Efficient Detection and Interpretation of Clusters in High Dimensional Databases

A Thesis

Submitted in fulfillment of the

requirements for the award of the degree of

Doctor of Philosophy

Submitted by

Mamta Mittal
(Registration No. 950803004)

Under the supervision of

Dr. R. K. Sharma
Professor
Computer Science & Engineering Department

Dr. V. P. Singh
Assistant Professor
Computer Science & Engineering Department

Computer Science & Engineering Department
Thapar University
Patiala -147004, Punjab, India
April 2015

CERTIFICATE

I, **Mamta Mittal** hereby certify that the work which is being presented in this thesis entitled **"Efficient Detection and Interpretation of Clusters in High Dimensional Databases"**, in fulfillment of requirements for the award of the degree of **DOCTOR OF PHILOSOPHY** being submitted in Computer Science and Engineering Department (CSED), Thapar University, Patiala, Punjab, is an authentic record of my own work carried out under the supervision of **Dr. R. K. Sharma** (Professor, CSED, Thapar University, Patiala) and **Dr. V. P. Singh** (Assistant Professor, CSED, Thapar University, Patiala).

The matter presented in this thesis has not been submitted either in part or full to any other University or Institute for award of any degree.

Date: April 30, 2015

(**Mamta Mittal**)
Signature of Candidate

This is certified that the above statement made by the candidate is correct to the best of our knowledge.

(Dr. R.K. Sharma)
Professor,
Computer Science & Engineering Department,
Thapar University, Patiala,
Pin-147004 (INDIA)

(Dr. V.P. Singh)
Assistant Professor,
Computer Science & Engineering Department,
Thapar University, Patiala,
Pin-147004 (INDIA)

Date: April 30, 2015

Acknowledgement

The real spirit of achieving a goal is through the way of excellence and austerous discipline. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by a range of personalities.

I have, indeed, been privileged to have worked under the guidance of Dr. R. K. Sharma (Professor, CSED, Thapar University, Patiala) and Dr. V. P. Singh (Assistant Professor, CSED, Thapar University, Patiala). I do not find adequate words to express my deep sense of gratitude towards them. Their personal guidance, encouragement, constructive criticism, invaluable feedback and stimulating discussion at all-time have been a source of inspiration to me in my work. This work has become possible only because of their priceless and unvarying efforts.

I extend my gratitude to Dr. Prakash Gopalan, Director, Thapar University, Patiala, for providing me an opportunity in the Thapar University, Patiala to carry out this research work. I also extend my heartiest thanks to the Doctoral Committee for monitoring the progress and providing priceless suggestions for improvement of my Ph.D. research work.

I am deeply grateful to Dr. Deepak Garg, Head CSED, Thapar University Patiala, for providing me university's resources and the necessary facilities for carrying out this work. I want to pay my special thanks to Dr. Seema Bawa (Former Head, CSED, Thapar University, Patiala) for introducing me with Dr. V. P. Singh.

I am also grateful to all academic, administrative and technical staff from the Thapar University, Patiala for their encouragement, timely assistance and acquaintance throughout my candidature.

I wish to express my profound gratitude to my family members, who supported me morally and emotionally. With the grace of God, this work provides me the opportunity to make my in-laws: Mr. Lajpat Rai and Mrs. Kamla Devi feel proud. I want to thank my adorable husband Mr. Lalit Mohan Goyal and my lovely son Ojas Goyal for providing a very loving, cool and calm environment in our home.

Mamta

Mamta Mittal

Abstract

Exponential growth of data resources has necessitated new techniques that can convert it into useful information. Clustering is one of the data mining techniques that investigates these data resources for hidden patterns. Many clustering algorithms are available in literature. This research work emphasizes on partitioning based methods and is an attempt towards developing clustering algorithms that can efficiently detect clusters for high dimensional databases. In partitioning based methods, *k*-means and single pass clustering are popular clustering algorithms but they have several limitations. To overcome the limitations of these algorithms, a *M*odified *S*ingle *P*ass *C*lustering (*MSPC*) algorithm has been proposed in this work. It revolves around the proposition of a threshold similarity value. This is not a user defined parameter; instead, it is a function of data objects left to be clustered. In our experiments, this threshold similarity value is taken as mean/median of the paired distance of all data objects left to be clustered. To assess the performance of *MSPC* algorithm, five experiments for *k*-means, *SPC* and *MSPC* algorithms have been carried out on artificial and real datasets.

Further, a deterministic algorithm, *A*daptive *T*hreshold based *C*lustering (*ATC*) has been proposed. It does not select the data objects randomly; rather, it is based on selecting the farthest data objects. It uses a parameter, neighborhood distance, to cluster the data objects. It is again an adaptive parameter and not specified by the user. Another parameter used in *ATC* algorithm is the minimum support value which prunes the insignificant clusters. Performance of the *ATC* algorithm is also assessed on ten artificial and eight real datasets. It has also been compared with existing *k*-means algorithm.

In this research work, new separation and compactness measures have also been proposed. Proposed compactness measures are based on the arithmetic/geometric average of maximum dispersion of data objects along each dimension. Proposed separation measure is an averaged paired distance between the data objects of clusters. Experimental work has been carried out on artificial and real datasets to justify these measures. The work presented in this thesis can be extended further by proposing variants in *MSPC* and *ATC* algorithms.

List of Figures

List of Tables

Contents

Chapter 1
Introduction

Knowledge discovery and data mining are two key areas that use data clustering to get insight into the distribution of data, *i.e.*, how the data is organized and how further meaningful information can be obtained by analyzing it. Therefore, it is necessary to have an overview of these areas with sufficient background details about clustering techniques. This chapter presents this overview, in brief. Various methods to validate clustering results are also illustrated in this chapter. In addition to it, the challenges, motivation and contribution of this research work have been described.

1.1 Background and Problem Statement

Today, every organization is dealing with data repository systems like relational databases, data warehouses, temporal databases, transactional databases, spatial databases, multimedia databases or the World Wide Web, but a lot of them are not able to take advantage of their huge repositories. Data to be stored is often diverse in nature ranging from scientific to medical, geographic to demographic, financial to marketing as well as the volume of data is so high that human analyst can not predict it without special tools. To automatically understand and analyze the data effectively and efficiently, the field of data mining has emerged in recent years. One of the data mining tools that can be used to group the data objects into unknown classes is clustering. The goal of clustering is to discover the natural grouping among data objects such that the data objects in the same group are similar to one another and dissimilar to the data objects in other groups. Intensive research has been carried out in this field and many algorithms have been proposed. But, clustering is an NP-hard problem due to which the existing approaches have some limitations. To deal with the limitations of existing methods, research is continuously being done in this area.

The problem statement of this research work is to detect and interpret the clusters in an efficient manner. Clustering is defined as: let $D = \{d_1, d_2, \dots, d_n\}$ be the dataset of n data objects such that each data object is in $\mathbb{R}^{\boldsymbol{d}}$. Clustering aims to group all the data objects into k $(1 < k \leq n)$ clusters represented as $K = \{k_1, k_2, \dots, k_k\}$ with $\bigcup_{i=1}^{k} k_i = D$, where k may be

unknown. Data objects are grouped into clusters such that each cluster has at least one data object, *i.e.*, $k_i \neq \emptyset, 1 \leq i \leq k$.

Further, clusters can be classified into the following categories based on whether they are allowed to overlap or not:

- Non-overlapped clusters: Each data object $d_i, 1 \leq i \leq n$, belongs to a single cluster, *i.e.*, $K_i \cap K_j = \emptyset, 1 \leq i, j \leq n, i \neq j$.
- Overlapped clusters: There exists at least one data object $d_i, 1 \leq i \leq n$, that belongs to more than one clusters.

1.2 Knowledge Discovery Process

*K*nowledge *D*iscovery in *D*ata (*KDD*) describes the process of extracting knowledge from a given database (Han and Kamber, 2006). In the context of knowledge discovery, knowledge means a pattern among the data objects. The term *data mining* is reserved exclusively for the discovery stage of *KDD* process. So, *KDD* is the non-trivial extraction of implicit, previously unknown, potentially worthy, and ultimately logical patterns from the databases. This process consists of six sub-processes as shown in the Figure 1.1.

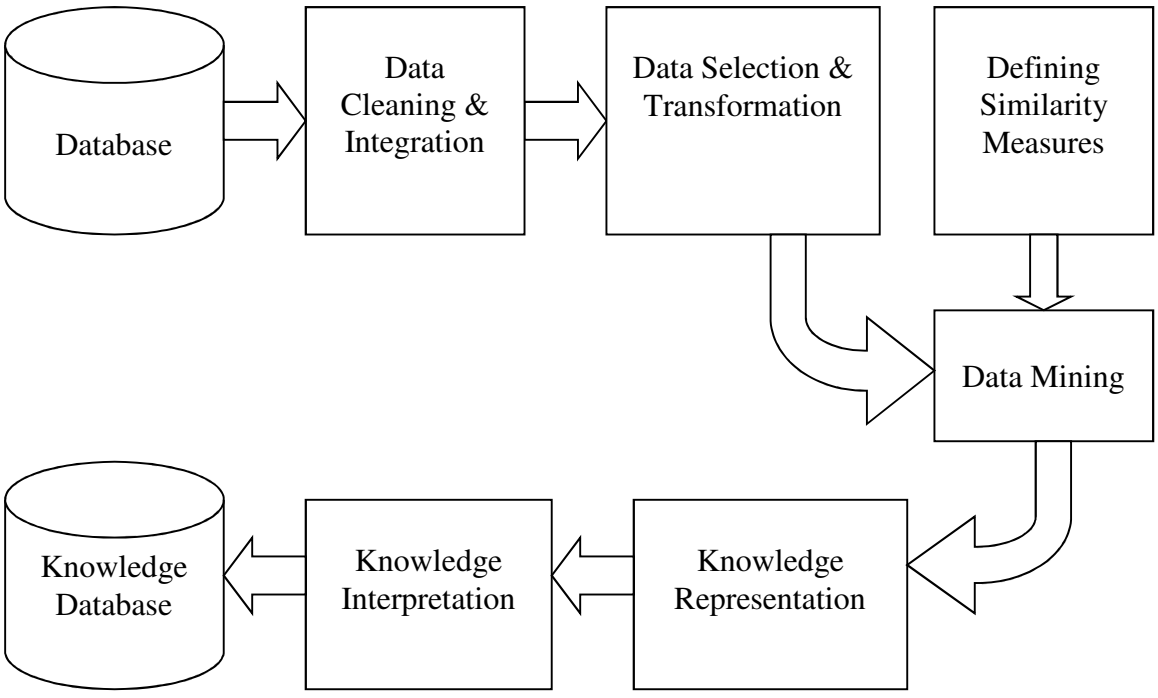


Figure 1.1: Knowledge discovery process

Data cleaning process removes the noise and inconsistencies from the databases. Cleaned databases from various resources are merged by the data integration process. As the size of database is huge and mining is a subjective task, data selection and transformation plays an important role in filtering scrap data. All these processes form the basis of data preprocessing. Preprocessed data is later analyzed by data mining techniques. It is possible for the data mining step to interact with the user or an already existing knowledge base. The user may expect to see some interesting patterns which may or may not be new within the knowledge base. According to this view, data mining contributes to the most essential sub-process of *KDD* process which reveals the useful hidden patterns for evaluation.

1.3 Data Mining

Data mining is used to discover knowledge from a large amount of data which is stored in information repositories such as databases, data warehouses, *etc.* Data mining stands at the intersection of techniques adopted from multiple disciplines such as statistics, high performance computing, image and signal processing, neural network, information retrieval, pattern recognition, machine learning and temporal data analysis where the focus is on efficient data mining techniques. Data mining is defined by Fayyad *et al.* (1996) as follows:

Data mining is the non-trivial extraction of implicit, previously unknown and potentially useful information from data.

1.3.1 Why is the data mining so popular?

Data mining is used to mine useful data from a large amount of data akin to the extraction of minerals from mineral ores. Most international organizations produce high amounts of information that could never be read by any person in a lifetime. The situation is even more alarming in the world wide networks. These days, gigabytes of data are distributed and exchanged over the world, the existing database management system allow retrieval of data but provide no tools to analyze it. Analysis is beneficial for unearthing the hidden relationships among the data. Data mining is one of the data analysis tools. It goes beyond the idea of conventional data analysis. It uses traditional analysis tools like statistics and graphics in conjunction with those associated with the field of artificial intelligence such as rule induction and artificial neural networks. It is an amalgam of all of these, but still somehow different.

Data mining is a distinctive approach towards the usual data analysis in the sense that the emphasis is not as much on extracting the facts as on generating the hypotheses. It is also capable of generating new business opportunities, the only condition being the provision of databases of sufficient size and acceptable quality. It is popular as it has the following capabilities:

- Prediction of trends and behaviors: It automates the tedious process of finding predictive behavior or information in huge databases. Before the emergence of data mining field, queries required excessive hands-on analysis, but now they can respond quickly and can be automated. Data mining analyzes past data to identify future trends. A common example is to know the future trends of the stock market.

- Automated discovery of previously unknown patterns: Data mining tools play a great role in identifying patterns which were previously hidden by sweeping through the database. A common example is to identify correlated products from the data of retail sales.

1.3.2 Architecture of data mining

The architecture of data mining system consisting major components are shown in Figure 1.2. In this figure, data cleaning, integration and selection techniques are performed on a single or a set of different databases. Database or data warehouse server makes sure that the fetched data is relevant and agrees with the user's request. A data warehouse stores the historical information which is helpful in taking the decisions strategically. Data mining engine is comprised of a set of functional modules for performing these tasks: clustering, association rule mining, classification, outlier analysis*, etc.* Many algorithms exist in the literature to perform these tasks efficiently. Data mining engines generates patterns helpful for erection of a knowledge base. Knowledge base component provides past knowledge to the data mining engines and pattern evaluation component. It stores current patterns which helps users to have updated knowledge. Pattern evaluation module interacts with the data mining engine to ensure that the discovered patterns are interesting and useful. This is initiated when the user specifies interestingness measures. Discovered interesting patterns are stored in the knowledge base. In the next section, various data mining techniques are discussed.

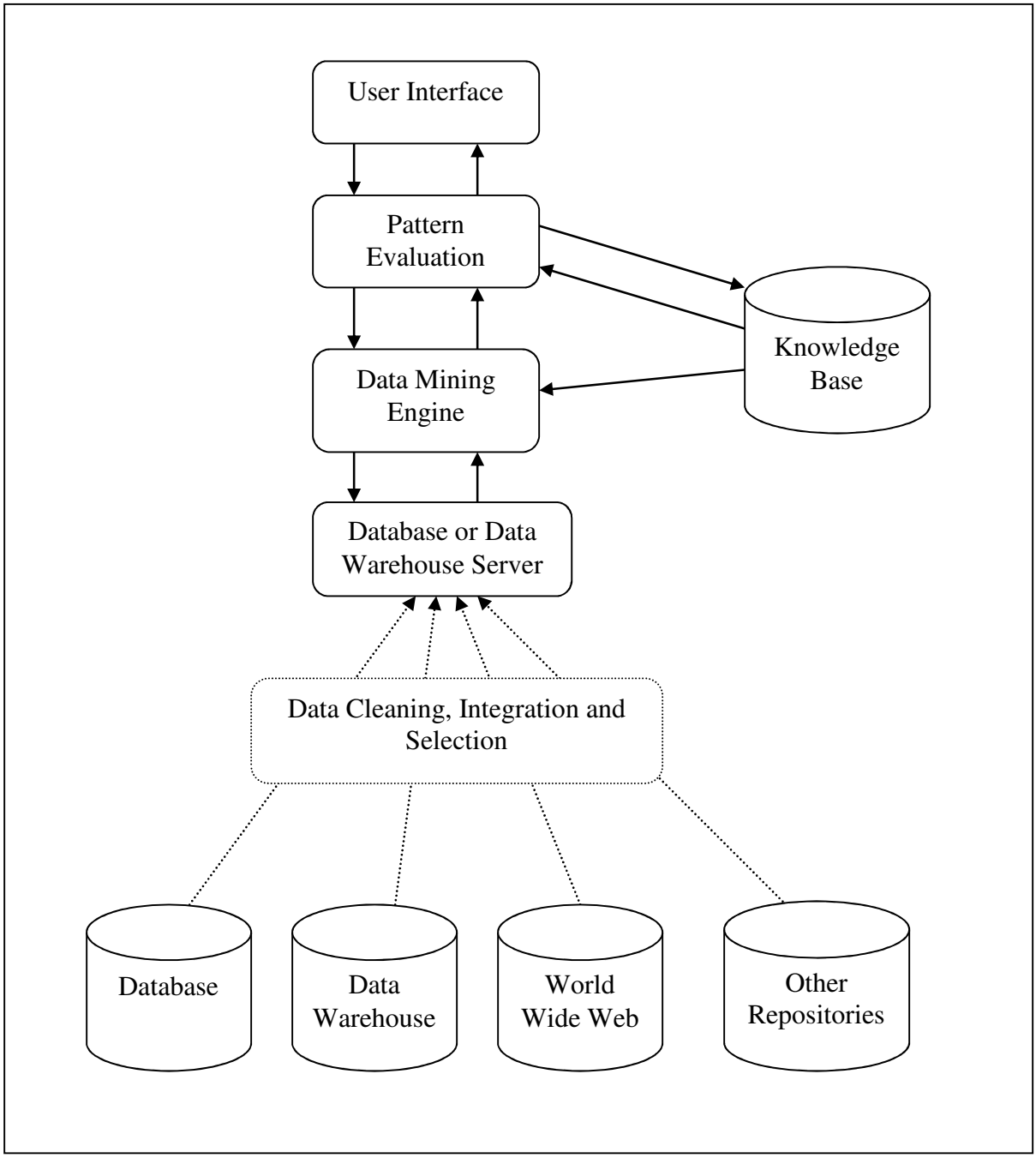


Figure 1.2: Architecture of data mining

1.3.3 Data mining techniques

The field of data mining encompasses mainly three techniques: association rule mining, classification and clustering as shown in the Figure 1.3. These techniques are explained briefly in sub-sections 1.3.3.1 - 1.3.3.3 below.

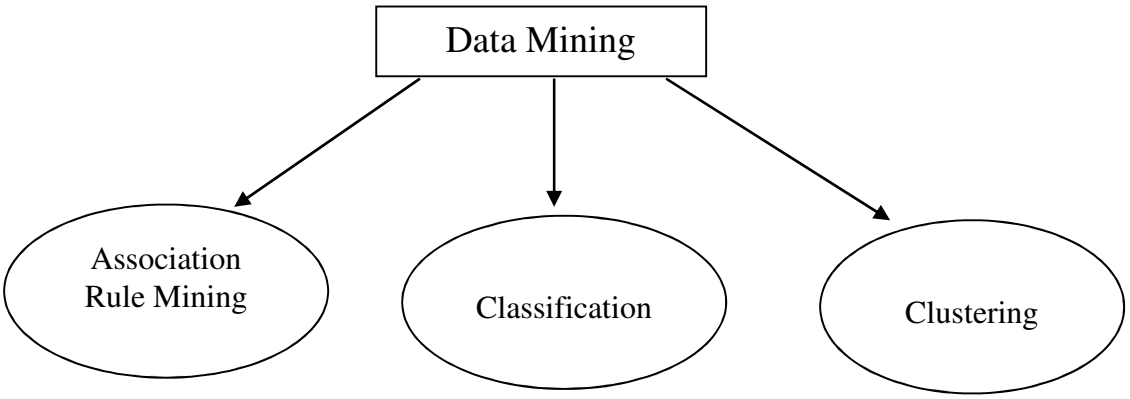


Figure 1.3: Data mining techniques

1.3.3.1 Association rule mining

*A*ssociation *R*ule *M*ining (*ARM*) generates an implication between two or more data objects of a database. In a transactional database for the given items, an association rule, $X \overset{m\_sup}{\Longrightarrow} Y$, is an implication where X and Y are disjoint sets of items; *m_sup* is the value of minimum support. The meaning of such an implication is that *m_sup%* transactions of a database which contain X also contain Y. For example, 90% (*m_sup%*) of the customers who purchase milk and eggs ($X = \{milk, eggs\}$) also purchase apples and onions ($Y = \{apples, onions\}$). In other words, the meaning of such rules is that *m_sup%* transactions of the given database have all the items of set X and all the items of set Y. The *m_sup* is a user defined parameter.

1.3.3.2 Classification

Classification is a supervised learning technique which classifies the data objects into pre-defined classes or categories. Its objective is to precisely classify each data object into a target class. For example, a new model of car can be put into one of the existing categories like hatchback, sedan. A classification model is derived based on the analysis of a set of training data. These models may be represented in various forms, like classification rules, decision trees or neural networks.

A well known classification problem is the binary classification, also known as IF-THEN rules. In this classification, the target attribute has only two possible values: low or high. In other type of classifications, target attribute has more than two values: very low, low, medium, high or very high. These types are represented by decision trees. Decision tree can easily be transformed to classification rules.

1.3.3.3 Clustering

Clustering, unlike classification, analyzes data objects without considering class labels. It is an unsupervised learning technique which groups the data objects into unknown classes (Chapelle *et al.* 2006). The grouping of data objects is based on the principle of maximizing the intra-class similarity and minimizing the inter-class similarity. As clustering does not use pre-defined class labels, it is distinct from classification which seek to find rules for classifying data objects into pre-defined classes. This research work is carried out in the field of clustering; hence, a brief overview is given in the next section.

1.4 Brief Overview of Clustering

Clustering is an *unsupervised learning* technique which concerns itself with finding a *structure* in a collection of unlabeled data. Clustering is a common data analysis technique used in many fields like pattern recognition, information retrieval, machine learning, *etc.* It is the segregation of dissimilar data objects into the different groups, so that the data objects share least common traits and it is the aggregation of similar data objects into the same group sharing most common traits. Figure 1.4 illustrates the concept of clustering with a graphical example.

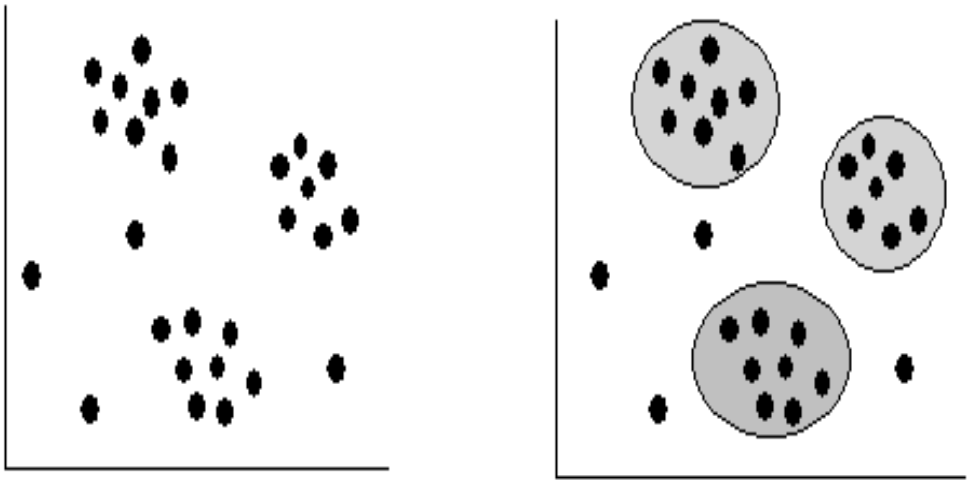


Figure 1.4: Clustering of data

In the above figure, most of the data objects are grouped into three clusters on the basis of similarity. The remaining data objects are identified as noise, *i.e.*, outliers, missing, unusual, or erroneous data. Hodge and Austin (2004) have given a survey on outlier detection

methods. The similarity between data objects depends on the *distance measure*. Data objects belong to same cluster if they are within distance limits from the representative of the cluster.

1.4.1 Clustering methods

A number of clustering methods exist in the literature. We may select a particular method on the basis of the type of output desired, facts regarding the performance of an algorithm with particular data, size of the database and the system configuration. These methods can be classified into the following categories.

1.4.1.1 Partitioning based methods

In the partitioning based methods, the underlying philosophy is to constitute *k* distinct clusters for *n* distinct data objects such that each data object is associated with atleast one of the clusters and that each cluster contains atleast one data object.

We have *k*-means, *S*ingle *P*ass *C*lustering and *P*artitioning *A*round *M*edoids (*PAM*) algorithms in this category. In these algorithms, *k* partitions are obtained for a given database where each partition represents a cluster. Data objects of each cluster are associated to the representative of the cluster known as centroid as shown in the Figure 1.5 with the circled star representing data objects and remaining star representing their centroids for three clusters.

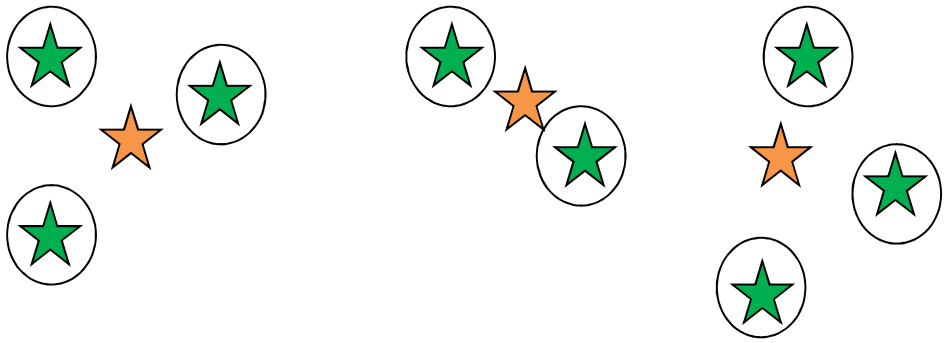


Figure 1.5: Partitioning based method

1.4.1.2 Hierarchical methods

These methods generate a hierarchical decomposition of the given data objects. There are two types of decomposition: agglomerative and divisive. The agglomerative approach, also known as bottom-up approach, generates a separate cluster for each data object and then

continues to merge the closer clusters. The process is successively repeated until a stopping criterion is reached. The divisive approach, also called the top-down approach, considers all data objects in a single cluster and fragments it into smaller clusters until a stopping criterion is reached. As shown in the Figure 1.6, agglomerative starts with distinct single object clusters, larger clusters are constituted by closest pair of clusters at each step.

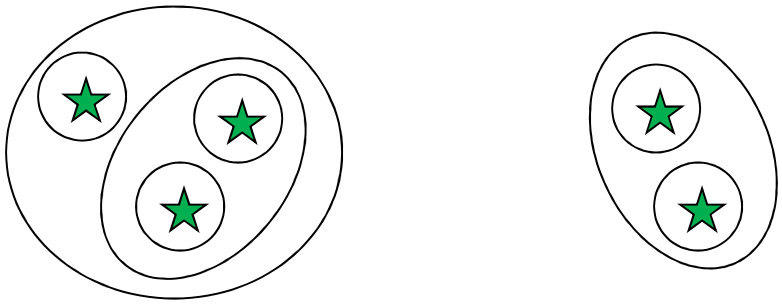


Figure 1.6: Hierarchical method using an agglomerative approach

The hierarchical and partitioning methods differ by either generating a nested series of partitions or disjoint partitions. In the hierarchical methods, small partitions are the subset of large partitions but in the partitioning methods, each partition is a disjoint set. Hierarchical methods produce good quality clusters when compared to the partition methods, but, partitioning methods defeats the hierarchical methods in terms of time complexity. The time complexity of partitioning methods is linear whereas it is quadratic for hierarchical methods. One can use both methods in conjunction also.

1.4.1.3 Density based methods

In density based methods, a cluster is a high density region of data objects distanced from low density regions as well as from other high density regions. In this method, cluster continues to grow until the density in the neighborhood exceeds some threshold value. Here, neighborhood and the threshold value are the user defined parameters. Neighborhood is the radius and threshold value is the minimum number of data objects within that radius. To find a cluster, start with an arbitrary data object *p* and retrieve the data objects in its neighborhood. If a data object contains the given minimum number of data objects in its neighbourhood then it is known as the core object. Let *p* be a core data object; formation of new cluster starts with *p* and its neighboring data objects as the members of the cluster; size of cluster grows when atleast one neighboring object of *p* found to be the core object. All

core objects identify new core objects in its neighborhood. Now, new core objects are also added as neighboring objects of *p* and the members of the cluster. Cluster grows till new core objects are identified. This process repeats for the remaining data objects.

1.4.1.4 Grid based methods

These methods distribute the data objects into a fixed number of cells in the form of a grid. The major benefit from this approach is its fast processing time independent of the number of data objects. In this method, data objects are mapped into the cells as shown in the Figure 1.7.

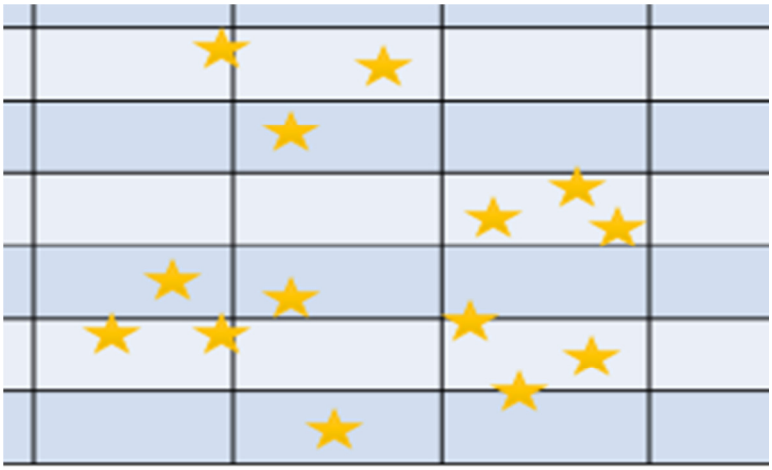


Figure 1.7: Grid based method

1.4.1.5 Fuzzy based methods

Fuzzy based methods are also known as soft clustering methods which are based on the philosophy that each data object can belong to more than one cluster and hence, we have a set of membership levels associated with each data object (Hoppner *et al.* 1999). Membership value is used to indicate how strongly an object is associated with a particular cluster. The most popular fuzzy algorithm is the Fuzzy c-means algorithm. It involves iterative optimization of a given objective function, in which membership of a data object and the cluster centers are updated during iteration.

1.4.2 Goals of clustering

The prime motive behind clustering is to determine the intrinsic grouping within a set of unlabeled data. There may be many different grouping results available but it might not be

possible to have the "best" among them. To resolve this, a user either selects a good clustering algorithm or provides the base for getting better clustering results based on user needs. For instance, the user may be interested in reducing the *data by keeping only the representatives of clusters*, getting familiar with unknown properties of clusters and detecting unusual data objects. By grouping the data objects into the clusters, their search and analysis improves significantly. The major goals of clustering are:

- To generate compact and distant groups.
- To reveal unknown properties among grouped data objects.
- To identify unusual data objects.
- To provide a representative for each group.

1.4.3 Characteristics of clustering algorithms

A clustering algorithm should possess the following characteristics (Cormen *et al.* 1990):

- Scalability: Scalability is the key issue in the implementation of clustering algorithm. These must be scalable for large as well as for high dimensional databases.
- Dealing with different types of attributes: Although a major proportion of algorithms are developed in sync with numerical data as input, it is necessary for the algorithms to accommodate other formats too. These could be binary numbers, spatial data, ordinal data, or even a mixture of two or more of these data types according to the application that the algorithm serves.
- Discovering clusters with arbitrary shape: A cluster's shape cannot be predetermined. Basic clustering algorithms based on the distance measure typically discover clusters to be of the same density and size with spherical boundaries. Though, this might not be true for every dataset. Hence, clustering algorithms should be designed in such a way that identify and work equally well with arbitrarily shaped clusters.
- Minimal requirements for domain knowledge to determine input parameters: In many algorithms there is a demand for the users to enter values of some predefined parameters for the algorithm to run and determine the clusters. These values supplied by the users can often be very crucial to how the end result of the clustering turns out.

This brings a lot of pressure on the users as the values should be accurate. In addition, it also puts the quality of the algorithm in jeopardy.

- Ability to deal with noise and outliers: Unlike ideal datasets that algorithms are designed to deal with, real world datasets are full of erroneous, missing, unknown data or outliers. Thus, algorithms should be resilient enough to handle such aberrations so that the quality of results generated is not compromised.
- Insensitivity to the order of input records: The manner or the sequence in which the data is given to the algorithm should not have an impact on the clustering results. Hence, it is crucial that the algorithm's behavior remains independent of the order in which it receives the input.
- High dimensionality: Algorithms, in general, work well with low dimensionality in the input datasets. Though, a database or a data warehouse in the real world is characterized by the presence of two or more attributes. Thus, the algorithms should be capable of successfully clustering multi-dimensional data.
- Constraint based clustering: Since real world models are implied by many constraints like domain knowledge, user given preferences. Clustering algorithms should also be adept to handle these constraints.
- Interpretability and usability: The results of a clustering algorithm should be easily represented, understood and interpreted by its users. The results generated are of not very useful if they cannot be successfully comprehended by the concerned parties. Thus, it is imperative to study the ultimate goal of an application before a clustering method can be employed.

1.4.4 Distance based similarity measures

A similarity value in the clustering means that the data objects are related closely in terms of the distance between them. There are following four measures to find similarity between two *d*-dimensional data objects $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ having integer or real values.

(i) City block or L_1 distance measure:

$$d_1(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sum_{k=1}^{d} \left| x_i^k - x_j^k \right|$$

(ii) Euclidean distance measure:

$$d_2(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sqrt{\sum_{k=1}^{d} \left| x_i^k - x_j^k \right|^2}$$

(iii) Minkowski distance measure:

$$d_p(\boldsymbol{x_i}, \boldsymbol{x_j}) = \sqrt[p]{\sum_{k=1}^{d} \left| x_i^k - x_j^k \right|^p}, p \geq 1$$

(iv) Cosine-correlation distance measure:

$$S_{cos}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{\sum_{k=1}^{d} \left(x_i^k x_j^k \right)}{\left(\sqrt{\sum_{k=1}^{d} (x_i^k)^2} \sqrt{\sum_{k=1}^{d} \left(x_j^k \right)^2} \right)}$$

These measures are explained below with the help of one example.

Let us take two 4-dimensional data objects $\boldsymbol{x_1} = (1, 0, 1, 0)$ and $\boldsymbol{x_2} = (2, 1, -3, -1)$. The four measures for these two data objects are given below.

(i) City Block or L_1 distance:

$$d_1(\boldsymbol{x_1}, \boldsymbol{x_2}) = 1 + 1 + 4 + 1 = 7$$

(ii) Euclidean distance:

$$d_2(\boldsymbol{x_1}, \boldsymbol{x_2}) = (1 + 1 + 16 + 1)^{1/2} = 4.36$$

(iii) Minkowski distance (for $p = 3$):

$$d_3(\boldsymbol{x_1}, \boldsymbol{x_2}) = (1 + 1 + 16 + 1)^{1/3} = 4.06$$

(iv) Cosine-correlation distance:

$$S_{cos}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{(2 + 0 - 3 + 0)}{(\sqrt{2}\sqrt{15})} = -0.18$$

Further, following two measures can be used to find similarity between *d*-dimensional data objects $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ having binary values. In these methods, a 2×2 contingency table, as given below, is used to find similarity measures in terms of Rand and Jaccard coefficients. Table 1.1 can thus be used to find the number of occurrences of *a*, *b*, *c* and *d* in the two *d*-dimensional data objects $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$. Let us denote these number of occurrences of *a*, *b*, *c*, *d* by A, B, C and D, respectively.

Table 1.1: Contingency table for calculating Rand and Jaccard coefficients

x_i \ x_j	1	0
1	*a*	*b*
0	*c*	*d*

(i) Rand Coefficient:

$$S_{rc}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{(A+D)}{(A+B+C+D)}$$

(ii) Jaccard Coefficient:

$$S_{jc}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{A}{(A+B+C)}$$

These measures are explained below with the help of one example.

Let us take two 8-dimensional data objects $\boldsymbol{x_1} = (0,0,1,1,0,1,0,1)$ and $\boldsymbol{x_2} = (0,1,1,0,0,1,0,0)$. For these data objects, the values of the A,B, C and Das per the contingency table are obtained as $A = 2$, $B = 2$, $C = 1$ and $D = 3$. The two coefficients thus shall be:

(i) Rand Coefficient:

$$S_{rc}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{5}{8} = 0.62$$

(ii) Jaccard Coefficient:

$$S_{jc}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{2}{5} = 0.40$$

1.4.5 Applications of clustering

There are many applications of data clustering in our everyday lives. Knowingly or unknowingly, we do perform clustering many times based on the similarities in our surroundings. In the field of computers, and specifically in information retrieval, it is noted that data clustering plays a pivotal role. Some such instances are as follows:

- *Biology*: Groups of genes characterized by a correlation in their respective expression patterns are formed from homogenous sequences by clustering them into gene families.

- *Medical:* In the field of medicinal sciences, clustering finds great utility in defining a distinctive boundary between various types of tissue and blood represented in a three-dimensional image. In addition to this, antibiotic resistances can also be traced as patterns so as to classify them on the basis of the antibacterial activity they carry out.

- *Business and marketing:* The information obtained through surveys and test panels are a valuable asset to market researchers who are forever trying to better understand their customers. This data is often multivariate and can aid in identifying the various segments into which different customers lie based on their buying trends. This information gives way to a better arrangement of shopping items such that it appeals to the consumer's sensibilities. In addition, it also provides great feedback for future development of new products and test market selection.

- *Insurance*: In fraud detection and to find the set of insurance policy holders who have a high average claim cost, clustering can be used.

- *City-planning*: On the basis of certain information parameters such as the value, location or type of houses they can be distributed into specific groups when clustered.

- *Earthquake studies*: The epicenters of previously occurred earthquakes can be studied by clustering algorithms if they are fed as the input data in order to produce, through observation, information on possible zones that are more likely to be afflicted by earthquakes.

- *Robotics Field*: Situational awareness in an essential characteristic that enables robots to track data objects and also in the detection of outliers in the data received by their sensors. This is greatly backed up by the technique of clustering.

- *Computer Science:* The huge array of computer science fields in which clustering finds great utility are marked by Markova chain Monte Carlo methods, evolutionary algorithms, image segmentation, the evolution of software, and many more.

- *Social Science:* Information as specific as the extensive levels of the occurrence of types of crimes in specific areas over a period of time can be extracted through clustering. This is specifically of use to the law enforcement bodies as it renders them more accurate information to work on. Also, even in the educational sector, the schools or student groups with similar characteristics or properties can be bifurcated through the application of clustering techniques.
- Climatology: Clustering can be employed as a successful technique for finding preferred sea level pressure through atmospheric pattern analysis, and also to trace out certain weather regimes.
- *WWW*: Document classification entails the grouping together of files in an intelligent manner. This requires clustering to be executed correctly. Web log data is also fed as input to clustering algorithms that aid in finding out similar and repeated access patterns.

1.5 Validation of Clusters

Clustering algorithms depend on the initial parameters in order to partition the datasets into clusters. Different clustering algorithms exhibit different results when applied on the same dataset, since they are very sensitive to the characteristics of original dataset, specially noise and dimensions. As a consequence, for the success of the clustering application, it is necessary that the results of the clustering algorithms be validated through some means. Various validation measures and indices determine the purity of the clusters. Purpose of the validation techniques is to express the quality of clusters, the degree with which a clustering scheme fits to a specific data set and the optimal number of clusters present in the databases.

1.5.1 Validation techniques

Cluster validation is quite difficult and challenging task in the field of clustering. Cluster validation techniques are divided into two groups: External and Internal. In external validation, the partitions are validated on behalf of the information available regarding the exact partition of a database. But, as we know, in real applications, grouping of the data object is not known previously. Therefore, the exact partitions are not available. If information about exact partitions is available, then Rand and Jaccard similarity measures are used. In the internal validation technique, external information is not available. Unlike the

external validation, internal validation relies on a structure hidden in the database. In this case, validity indices Dunn and *D*avies-*B*ouldin (*DB*) are mostly used.

1.5.2 Validity measures

Validity indices used in the internal validation techniques are defined by two validity measures: Compactness and Separation. Compactness is the measure of intra-cluster distances of the data objects; it should be as minimum a value as possible. It is also called as the measure of similarity of data objects in a cluster. Separation is the measure of inter-cluster distance between clusters; it should be as maximum a value as possible. It is also called as the measure of similarity among data objects of two clusters.

1.5.2.1 Compactness

The compactness of a cluster C having *d*-dimensional *m* data objects can be measured by the following two methods. In these methods, x_i^k is the value of k^{th} dimension of data object $\boldsymbol{x_i}$ and x_j^k is the value of k^{th} dimension of data object $\boldsymbol{x_j}$ in cluster C.

(i) Centroid Linkage:

$$Comp_{cen}(C) = \frac{\sum_{i=1}^{m}\sqrt{\sum_{k=1}^{d}(x_i^k - x_c^k)^2}}{m}$$

where, $\boldsymbol{x_c}$ is the *d*-dimensional centroid of cluster C

(ii) Averaged paired distance:

$$Comp_{avg}(C) = \frac{2}{m(m-1)}\sum_{x_i^k \in C,\ x_j^k \in C}\sqrt{\sum_{k=1}^{d}(x_i^k - x_j^k)^2}$$

1.5.2.2 Separation

Separation between clusters C_i and C_j can be measured by the following three methods. In these methods, x_i^k is the value of k^{th} dimension of data object $\boldsymbol{x_i}$ belonging to cluster C_i and x_j^k is the value of k^{th} dimension of data object $\boldsymbol{x_j}$ belonging to cluster C_j.

(i) Single linkage: It is the distance between two closest data objects of clusters C_i and C_j.

$$Sep_{min}(C_i, C_j) = \min_{x_i^k \in C_i,\ x_j^k \in C_j}\left(\sqrt{\sum_{k=1}^{d}\left(x_i^k - x_j^k\right)^2}\right)$$

(ii) Centroid linkage: It is the distance between the centroids of clusters C_i and C_j.

$$Sep_{cen}(C_i, C_j) = \sqrt{\sum_{k=1}^{d}(x_{c_i}^k - x_{c_j}^k)^2}$$

where, $\boldsymbol{x_{c_i}}$ and $\boldsymbol{x_{c_j}}$ are *d*-dimensional centroids of cluster C_i and C_j, respectively.

(iii) Complete linkage: It is the distance between two farthest data objects of clusters C_i and C_j.

$$Sep_{max}(C_i, C_j) = \max_{\substack{x_i^k \in C_i, \\ x_j^k \in C_j}} \left(\sqrt{\sum_{k=1}^{d}(x_i^k - x_j^k)^2} \right)$$

1.5.3 Validity indices

The validity indices combine the separation and compactness measures for evaluating the quality of clustering algorithms. This combination may be a ratio or a summation of the measures. Two validity indices, namely, Dunn and *D*avies-*B*ouldin (*DB*) have been used in the present research work.

1.5.3.1 Dunn index

Dunn (1974) defined an index as a ratio of the separation to the compactness measure. In this index, separation is measured by the single linkage method and the compactness is measured by determining the distance between the farthest data objects of a cluster. Mathematically, it is defined as:

$$dunn_n = \min_{i=1,\ldots,n}\left\{\min_{j=i+1,\ldots,n}\left(\frac{dist(C_i, C_j)}{\max_{k=1,\ldots,n} diam(C_k)}\right)\right\}$$

where, *n* is the number of clusters, $dist(C_i, C_j)$ is the measure of separation between clusters C_i and C_j and $diam(C_k)$ is the measure of compactness between the farthest data objects of a cluster C_k.

1.5.3.2 *D*avies-*B*ouldin (*DB*) index

*D*avies-*B*ouldin (1979) defined an index as a ratio of compactness to the separation. In this index, similarity of a cluster with respect to other cluster is defined. It is the ratio of compactness of both clusters to the separation between them. Mathematically, similarity of cluster C_i with respect to cluster C_j is defined as:

$$R_{ij} = \frac{S_i + S_j}{d_{ij}}$$

where, S_i and S_j are the centroid based compactness of clusters C_i and C_j; and d_{ij} is the centroid linkage based separation between them. Maximum similarity value of each cluster is averaged to get the value of *D*avies-*B*ouldin (*DB*) index. Mathematically, it is defined as:

$$DB_n = \frac{1}{n}\sum_{i=1}^{n} M_i$$

where, M_i is the maximum value of R_{ij} for the cluster C_i and n is the number of clusters.

1.6 Challenges and Motivations

During the last several decades, many clustering algorithms have been proposed by researchers, yet there exists some open challenges in the clustering as follows:

- Clustering results strongly depend on the selection of the user defined parameters. For example, the number of clusters in *k*-means algorithm and threshold value in *S*ingle *P*ass *C*lustering algorithm are user defined parameters.

- In the real datasets, noise or outliers often exist due to erroneous measurement, or human error. These outliers have a great influence on the clustering results. So, new methods must be able to detect and remove outliers.

- Nowadays, a large amount of datasets are available in biology, neuroscience, information retrieval, multimedia and spatial applications area. Data in these fields are generally complex in nature. To deal with such data, specific similarity measures are required. However, defining new similarity measures for handling complex data is a complex task.

- Many clustering algorithms generate the spherical shaped clusters but clusters may be of arbitrary shapes.

- Clustering algorithms either generate overlapped or non-overlapped clusters. An approach which has ability to generate both types of clusters may be required.

- Non-deterministic approaches produce varying results on successive runs for the same dataset. This motivates researchers to work on the deterministic approach.

These challenges motivated us to propose clustering algorithm that focus on deterministic approach; does not require user defined parameters during clustering process; generates both overlapped and non-overlapped clusters and detects outliers.

1.7 Objectives of this Research Work

Following objectives were identified when this research work was initiated.

- To explore and analyze techniques for clustering in high dimensional databases.
- To propose, design and implement algorithms for clustering in high dimensional databases.
- To test and validate the proposed algorithms through case study.

1.8 Datasets used in the Experiments

In this research work, eight real high dimensional datasets have been used (Table 1.2). These datasets are taken from *U*niversity of *C*alifornia, *I*rvine (*UCI*) repository.

Table 1.2: Real datasets used in the experiments

S. No.	Dataset	# Data objects	# Dimensions
1.	BCW-O	683	9
2.	Ecoli	336	7
3.	G.I.	214	9
4.	Haberman's Survival	306	3
5.	Iris	150	4
6.	Seeds	210	7
7.	Wine	178	13
8.	Yeast	1484	8

1.9 Contribution and Applications of this Research Work

The main objective of this research work has been to detect and interpret clusters in high dimensional databases. In order to achieve the objectives, comprehensive review of the literature on various clustering algorithms and validation techniques has been carried out.

Clustering algorithms such as *k*-means and *S*ingle *P*ass *C*lustering have been discussed for partitioning the datasets. A detail discussion is presented about them by the experiments carried out on the artificial and real datasets. Performance of both algorithms is also compared using validity measures and indices. The experimentation carried out on them uncovers that *S*ingle *P*ass *C*lustering algorithm performs better than *k*-means algorithm for the popular validity measures and indices. Limitations of both algorithms are also discussed.

To overcome the limitations of *k*-means and *S*ingle *P*ass *C*lustering algorithms, we have proposed a *M*odified *S*ingle *P*ass *C*lustering (*MSPC*) algorithm. In this algorithm, the concept of a variable threshold value is presented which is not a user defined parameter. Rather, it is a function of data objects left to be clustered. This algorithm generates the clusters automatically. Some experiments have been performed for *k*-means, *SPC* and *MSPC* on real datasets: *Ecoli, Iris, Seeds* and *Wine* and also on artificial datasets. The quality of clustering is assessed by means of separation and compactness measures. The experiments carried out in this work reveal that *MSPC* algorithm generates actual number of clusters present in the dataset and performs better than *k*-means and *SPC* algorithms. It is also observed that the *MSPC* algorithm works efficiently for high dimensional datasets.

Further, a novel *A*daptive *T*hreshold based *C*lustering (*ATC*) algorithm has been proposed which uses neighborhood distance value to cluster high dimensional datasets. In this algorithm, neighborhood distance value is not a user defined parameter; rather, it is automatically calculated and makes the proposed approach adaptive. Performance of the novel adaptive clustering algorithm is evaluated and compared with *k*-means algorithm on artificial and real datasets. Some key observations noted from the experiments are: (i) It is deterministic in nature, it gives same results on successive runs on a dataset (ii) It automatically generates the clusters (iii) It detects outliers (iv) It generates overlapped and non-overlapped clusters and (v) *k*-means algorithm partitions the dataset into *k* clusters of almost similar size whereas proposed algorithm generates clusters of different sizes.

Further, new compactness and separation validity measures have also been proposed in this work. Experimental work is performed on artificial and real datasets to justify these measures. It is observed that proposed compactness and separation measures can be used as alternative validity measures in this field of clustering.

The research carried out in this thesis will help in the field of medical science, criminal investigation, agriculture and bioinformatics.

1.10 Organization of Thesis

This thesis is divided into six chapters. In the current chapter, the fundamentals related to the problem have been introduced. In the second chapter, a comprehensive review of the literature on various methods used for partitioning based clustering techniques has been presented. The third chapter demonstrates the comparison of *k*-means and *S*ingle *P*ass *C*lustering algorithms. In this chapter, we have also proposed a *M*odified *S*ingle *P*ass *C*lustering (*MSPC*) algorithm. In the fourth chapter, a novel *A*daptive *T*hreshold based *C*lustering (*ATC*) algorithm is proposed which uses neighborhood distance value to cluster the high dimensional datasets. In the fifth chapter, new compactness and separation measures are proposed for measuring the quality of clustering algorithms. The last chapter of this thesis presents the conclusions drawn from the results of the experiments carried out in this work. Some indicators for future research on the topic under consideration are also discussed briefly along with the benefits and limitations of this research.

Chapter Summary

In this chapter, knowledge discovery process with introduction to data mining and its techniques: association rule mining, classification and clustering has been discussed. Association rule mining is a technique in which implication among the data objects are identified. In the classification technique, data objects are partitioned into predefined groups whereas in the clustering technique data objects are partitioned into undefined groups. Goals of clustering, its characteristics and popular validity measures are also described briefly in this chapter. Further, challenges and motivation of the work have also been presented. The metadata on the datasets used in this work is also presented in this chapter.

Chapter 2
Literature Review

Today's age of globalization has revolutionized the organizations and given a thrust to maintain humongous amount of data received from various resources. The rate of growth of these resources is so high that it is very difficult for organizations to focus on the relevant information content. Development of algorithms for the automatic extraction of the relevant information from the datasets has been a dream of human beings since a long time. People had been working for the realization of this dream even before the emergence of data mining. The works of MacQueen (1967) and Lloyd (1982) are noteworthy in clustering, one of the data mining techniques. They developed *k*-means clustering algorithm that partitions the datasets into undefined classes. A comprehensive review of the literature on the data mining, clustering techniques, partitioning based clustering techniques and validation techniques has been presented in this chapter.

2.1 Introduction

The exponential growth of data has necessitated new techniques that can convert it into the useful information. To collect this information, an organization has to execute a knowledge discovery process. Knowledge discovery process is also known as *K*nowledge *D*iscovery in *D*atabases (*KDD*). It is a data reduction process which prunes the data into the useful information known as knowledge (Fayyad *et al.*, 1996). It follows a streamlined process to identify non-trivial, implicit, understandable and useful patterns from the complex and large datasets. This process consists of multiple steps that are iterative. Data mining is included as a core step in the *KDD* process.

2.2 Data Mining

In 1960s, statisticians used the terms like "Data Fishing" or Data Dredging" to retrieve the information from the datasets. They proposed number of algorithms to partition the datasets into disjoint sets before the term "Data mining" emerged. Like in 1967, MacQueen has proposed *k*-means algorithm which is a multi-pass algorithm for obtaining disjoint sets. After that, Salton (1971) introduced single pass algorithm to partition the datasets. Duda and Hart

(1973) described the importance of partitioning the datasets in the field of pattern recognition.

In 1975, Bentley introduced the multi-dimensional binary search tree to retrieve information. In this, *k-d* tree is used as a data structure to handle different types of queries. It stores the information and separates the data objects into predefined categories. Dubes and Jain (1976) suggested the guidelines about the clustering techniques to users. They emphasized that the user should be aware of intrinsic properties of a clustering technique before using them. Friedman *et al.* (1977) presented an algorithm using the *k-d* tree data structure for searching the *k* best matches for a given query when the exact match is not available in the datasets. In 1979, a theoretical framework for information retrieval was given by Rijsbergen based on a non-standard logic. In order to characterize the uncertainty associated with any logical implication a new logical uncertainty principle was stated.

In 1982, Lloyd proposed *k*-means algorithm for partitioning the datasets into *k* disjoint sets. In these disjoint sets, distance between the data objects and their centroids is minimum. Alternately, Kaufman and Rousseeuw (1987) proposed *k*-medoid algorithm which is based on the search of *k* representatives called medoids in a given dataset. These medoids partition the dataset into *k* disjoint sets. In these sets, distance between data objects and their medoid is minimum. In 1993, Quinlan introduced the C4.5 algorithm that generates a classifier and expressed it in the form of decision tree. This decision tree is based on the attribute's information gain which splits the dataset into various classes.

Agrawal and Srikant (1994) initiated the field of association rule mining. They proposed a well-known apriori algorithm to generate frequent itemsets. It consists of two steps: generation of candidate itemsets and their verification in the datasets. Further, generation step includes two sub steps: joining and pruning. In the joining step, a pair of frequent itemsets is joined to generate candidate itemsets and in the pruning step, these itemsets are pruned. In the verification step, remaining itemsets are counted and compared against the given *min_sup* value. It iteratively performs these steps until all itemsets are generated.

Wang *et al.* (1997) proposed *ST*atistical *IN*formation *G*rid based (*STING*) algorithm. In this algorithm, a dataset is framed into rectangular cells known as grid. Multiple levels of rectangular cells are formed corresponding to levels of resolution. Execution time of this

algorithm depends on the number of cells in each dimension rather than the number of data objects. Traditional statistical learning was based on theoretical analysis until learning algorithms were developed. Vapnik (1999) provided an overview of these learning algorithms. Jain *et al.* (1999) have presented an exhaustive survey on the clustering techniques. *CL*-Tree (*CL*ustering based on decision Trees) proposed by *Liu et al.* (2000) is based on a supervised learning technique. Virtual data points are introduced into the space followed by the application of a modified decision tree algorithm. The proposed *CL*-Tree technique consists of two steps: construction of cluster tree and its pruning. For the construction of cluster tree, a modified decision tree algorithm is used to construct the tree with a new purity function; without making any prior assumption. Consequently, it captures the natural distribution of the data. After an interactive pruning step, meaningful clusters are identified in the form of hyper-rectangular regions.

Mitra *et al.* (2002) proposed an unsupervised feature selection algorithm which has the capability of representing the given datasets on multiple scales. They also proposed a new feature similarity measure. Minz *et al.* (2005) improved the decision tree classifiers using rough set tools. These tools identify the principal attributes to construct the classifiers. Berkhin (2006) has given an exhaustive survey on the field of clustering techniques and Han *et al.* (2007) have presented a survey on association rule mining.

In December 2008, top 10 algorithms are identified at the IEEE *I*nternational *C*onference on *D*ata *M*ining (*ICDM*) meeting (Wu *et al.*, 2008). These algorithms are being used by the research community in the field of classification, integration mining, rough sets, link mining, clustering, bagging and boosting, statistical learning, sequential patterns, *etc*. The identified algorithms are C4.5, *S*upport *V*ector *M*achine (*SVM*), EM, *k*-means, Apriori, *k*NN, PageRank, Naïve Bayes, AdaBoost and CART. These algorithms remain to be the most influential algorithms in the field of data mining that continue to engage researchers.

2.3 Clustering

Clustering is an important data mining technique to partition the datasets. Research work in the field of clustering has been going on continually throughout the world since late 1960's. It is an active research area even these days as the problem is complex in nature. In this section, survey on various clustering methods has been done.

2.3.1 Partitioning based clustering

Standard *k*-means algorithm was being used in Bell Labs since 1957 until published by Lloyd in 1982. It is widely accepted in the research community owing to its simplicity. This algorithm is an active research area since researchers encountered the following problems in this algorithm.

- How to know the number of clusters present in the datasets?
- How to provide the initial centroids for clustering?
- How to overcome the affect of outliers on the quality of clustering?

2.3.1.1 Determining the number of clusters

Knowledge about the dataset is a major stimulant for finding the number of clusters that could exist in it. One such algorithm that determines the number of clusters has been proposed by Ray and Turi (1999). This algorithm is applied to the segments of color images and measured the compactness and separation for different values of *k*. They have inferred that the minimum ratio of compactness to the separation produces the actual number of clusters present in the image.

In order to determine the number of clusters, a gap statistic was introduced by Tibshirani *et al*. (2001). This gap statistic is a measure of difference between observed intra-cluster to the expected intra-cluster distance. This minimum difference determines the optimal value of *k*. The gap statistic measure has been replaced by a new function given by Pham *et al.* (2004). This function measures the ratio of the real distortion to the estimated distortion for a uniform distribution of the data objects. The minimum value of this function determines the number of cluster present in the dataset. The gap statistic measure compares the dispersion of the clustered data objects but it may overestimate the number of clusters. Inspired from the gap statistic method, weighted gap and data distribution weighted gap methods have been proposed by Yan (2005). In the weighted gap method, averaged pair wise distance among all the data objects of a cluster is considered. This method may also overestimate the number of clusters, yet gives better results than the gap statistic method. The data distribution weighted gap method searches for an optimal number of clusters so that the observed gap is sufficiently small under suitable reference distribution.

McCullagh and Yang (2008) have proposed an improved Dirichlet allocation model to determine the number of clusters. Wang *et al.* (2009) has determined the number of clusters by creating the *R*eordered *D*issimilarity *I*mage (*RDI*) of an image dataset. *RDI* analyzes the clustering tendency of the images. It is helpful in identifying the clusters in the form of dark blocks that lie along the diagonal of the image corresponding to the dense areas of the image. Thereafter, a black and white image is created followed by filtering, transforming and projecting the pixel values on to the diagonal axis of the image. The first order derivative is computed for the diagonal axis of the image and the major peaks and valleys are used to decide the number of clusters.

2.3.1.2 Centroids initialization methods

In *k*-means algorithm, the problem of initialization of centroids has been solved by Katsavounidis *et al.* (1994). They considered the very first centroid in one corner of the dataset having highest norm. Subsequently, other centroids are selected as farthest data objects from the data objects selected so far. In this process, the number of centroids selected is equal to the number of clusters required in the datasets. Bradley and Fayyad (1998a) proposed a sampling method to initialize the cluster centroids. Data samples are clustered using *k*-means algorithm for the given value of *k*. After that, centroids of all the samples are clustered again for the same value of *k*; and now, the obtained centroids act as the initial centroids for the entire dataset. Likas *et al.* (2003) presented a global *k*-means algorithm. In this algorithm, initial centroids are obtained dynamically one at a time until number of centroids is equal to the value of *k*. First initial centroid is obtained by executing the *k*-means algorithm for $k = 1$. In this case, the mean value of the dataset acts as the first initial centroid. Second initial centroid is obtained by executing the *k*-means algorithm for $k = 2$. In this case, each data object is considered a temporary initial centroid until better partition is acquired. Subsequently, *k* centroids are obtained by executing the *k*-means algorithm for required value of *k*.

Khan and Ahmad (2004) have proposed an algorithm for initialization of cluster centroids. This algorithm focused on the values of individual attributes. The mean and standard deviation of each attribute is computed to calculate importance of an attribute in terms of percentile. Then, attributes are partitioned by executing the *k*-means algorithm. Each data

object is labeled according to the clustering results obtained from these partitions and a string of labels is generated corresponding to each attribute. Data objects with the same strings are merged into a single cluster. If the number of clusters generated is more than the required number of clusters then density based data compression is used to get the required number of clusters. Finally, the mean value of *k* clusters is used as *k* initial centroids for clustering the data objects in full dimensional space. Yuan *et al.* (2004) proposed a method to form a group of similar data objects with the help of adjacency matrix. Each group consists of fixed number of data objects. Initially, in the first group, two closest objects are assigned. Other objects are added in the group that has closest distance to any member of the group till number of data objects is equal to the given fixed value. Afterwards, other *k*-1 groups of similar data objects are generated dynamically from the remaining data objects. Mean of each group is considered as the initial centroids.

Redmond and Heneghan (2007) proposed the initialization method for *k*-means clustering algorithm using *k-d* tree. The dataset is partitioned into smaller partitions till either *k* number of partitions are obtained or the partitions contain a predefined number of data objects. Centroids of the partitions are obtained according to the partition axis perpendicular to the highest variance derived by principal component analysis. These centroids act as initial centroids for the entire dataset. Arthur and Vassilvitskii (2007) extended the work of Katsavounidis *et al.* (1994). In this method, the centroids are selected probabilistically, except the first centroid. The probability of their selection is proportional to their distance from the selected data objects. Deelers and Auwatanamongkol (2007) considered all the data objects in a single cell. Then, a data object splits the cell into two cells. This data object minimizes the squared Euclidean distance of all data objects present in the cells. Subsequently, cells are partitioned until *k* cells are formed. Now, centroid of each cell acts as initial centroids for the entire dataset.

Chiang and Mirkin (2009) extracted anomalous centroids by considering origin as a reference point. These centroids are used to form clusters. Small sized clusters are removed and the remaining clusters are the true clusters present in the dataset. The centroids of the remaining clusters act as initial centroids. Cao *et al.* (2009) used the rough set model to measure the cohesion degree of the nearest object and coupling degree between the neighborhood objects. The *k* objects which have maximum cohesion degree but some fixed coupling degree are

selected as initial centroids. Li (2011) proposed a centroid initialization method for the datasets having two clusters. In this method two nearest neighbor pairs are considered that are most dissimilar. These pairs should not be in the same cluster and moreover, should not be on the partitioning boundary of clusters. The mean of such selected pairs is considered as the initial centroids. Erisoglu *et al.* (2011) proposed a neighborhood model for initialization of cluster centroids. The mean object of the entire dataset is determined. Then, the data object farthest from the mean object is selected as the first initial centroid; second initial centroid is selected which is farthest from the first initial centroid; subsequently, the k^{th} initial centroid is selected at the distance farthest from the $(k\text{-}1)^{th}$ initial centroid. Data objects once selected as the initial centroids are not considered again.

Reddy and Jana (2012) have given a novel method to select the initial centroids with the help of the Voronoi diagram. It is formed from the given set of data objects. The initial centroids will be those data objects which lie on the boundary of the Voronoi circles having highest radius. Zhang and Cheng (2013) proposed an initialization method based on the density. Density of all data objects is determined and the data objects with the same density are considered in the homogeneous group. Representative of each group acts as the initial centroid. Celebi *et al.* (2013) have given an exhaustive survey on various methods to initialize the centroids.

2.3.1.3 Outlier detection methods

The outlyingness factor for each data object has been proposed by Hautamaki *et al.* (2005). It is inversely proportional to the in-degree of the data object. *k*-means algorithm is applied on a given dataset; after convergence, the value of the outlyingness factor is computed for each data object. The calculated outlyingness factor value is compared with the given threshold value to detect the outliers. The outliers are removed from the dataset and the *k*-means algorithm is applied again on the reduced dataset with the same initial centroids.

Hasan *et al.* (2009) proposed an algorithm named *ROB*ust *IN*itialization (*ROBIN*) which restricts outlier to participate in the selection of initial centroids. The density of each data object is calculated on the basis of the number of neighboring data objects within the given threshold value. The average relative density of each data object is measured as the ratio of the data object's density to the averaged neighboring data objects' density. A local outlying

factor is computed as the inverse of relative density value. Local outlying factor of each data object is compared with the local outlying factor of the neighboring data objects and if the outlying factor of data object is larger, then it is declared as an outlier. Jiang *et al.* (2009) proposed distance based algorithm to remove the outliers using rough set theory. In addition to it, the distance metrics required for this algorithm have also been discussed. Chen *et al.* (2010) proposed a neighborhood model to detect the outliers. This model also employs rough set theory and a neighborhood metric for the outliers. Kim *et al.* (2011) used *k-d* tree indexing and *k*-nearest neighbor search algorithm to reduce the computation time of local outlying factor. Local outlying factor is used to detect the low density data objects surrounded by high density data objects. These data objects are declared as outliers.

Yang and Zhu (2010) proposed a rough set based outlier detection method. In this method, outliers are detected in subset of dimensions instead of full dimension space. Key attribute are mined to detect the outliers by defining the outlying partition similarity. These aspects of *k*-means algorithm have, however, not restricted the use of this algorithm, rather motivated researchers for improving its functioning. A number of variants are available in the literature. Some of them have been discussed as below.

2.3.1.4 Variants of *k*-means algorithm

McLachlan and Krishnan (1997) provided a detailed description on the *E*xpectation-*M*aximization (*EM*) methods and its extensions. It is an iterative method for finding the maximum likelihood of parameters in the statistical model. It performs *E*xpectation (*E*) steps for estimating the probabilities of log likelihood and *M*aximization (*M*) steps for maximizing the expected log likelihood on the *E* steps. It is applied to continuous and to categorical variables.

An extended *k*-means algorithm named as *k*-mode was proposed by Huang (1998). This algorithm is employed to cluster the large datasets with numerical and categorical values. The similarity between data objects is calculated on the basis of proposed cost function. This cost function depends on the number of mismatches in the data objects. He used weighted sum Euclidean distance measure for mixed type of attributes, the weights must be known in advance. He has also argued that improper weights may yield the biased results. Pelleg and Moore (1999) presented that *k-d* trees can be used to reduce the number of nearest neighbor

queries. In this, the computation requirement is reduced greatly while updating the centroids through geometrical analysis. Pelleg and Moore (2000) proposed a method which does not require the exact value of *k*. Instead, a lower and an upper bound on the value of *k* are required. Initially, *k*-means algorithm is executed for lower bound value; after convergence, the value of *k* is increased by one. Increment in the value of *k* depends on two possibilities; either only one or each cluster is split into two clusters. In the former case, splitting is continued until *k* number of clusters are generated and in the latter, *2*-means is executed for each cluster until the desired value of *k* is achieved. The value of *k* which has best score determines the number of clusters.

Savaresi and Boley (2001) evaluated the performance of bisecting *k*-means and *P*rincipal *D*irection *D*ivisive *P*artitioning (*PDDP*) algorithms. Performance of both algorithms is compared and highlighted the similarity and differences between them. Kanungo *et al.* (2002) analyzed and implemented the *k*-means clustering algorithm. They used *k-d* tree as a major data structure to implement it. They proposed the idea to maintain candidate centers for each node of the *k-d* tree. These candidate centers are filtered while traversing the nodes. This implementation runs faster than the general implementation. The *k**-means algorithm has been proposed by Cheung (2003) as a generalized *k*-means clustering algorithm. Here, *k** is an indication of the actual number of clusters present in the dataset. This algorithm is helpful for ellipse-shaped clusters. It clusters datasets without asking the number of clusters and it is not sensitive to outliers. In this algorithm, at least one initial centroid is presented for each cluster and the data objects are adjusted adaptively by a learning rule proposed by him.

Fahim *et al.* (2006) proposed a variant of *k*-means algorithm. It improves the implementation of standard *k*-means algorithm by considering two data structures- first is used to store the index and the second is used to store the Euclidean distance. During iterations, previous distance of each data object is compared with the distance to its updated centroid. For an object, if previous distance is less than the new distance then the object is clustered with updated centroids. Ahmad and Dey (2007) extended the work done by Huang (1998). They proposed a modified cost function. In this function, significance of a categorical attribute is not user defined parameter; rather, it is inherent in the given dataset and distance function for categorical attributes does not use binary or discrete measures. It is computed as a function of overall distribution and co-occurrence with other attributes.

Zalik (2008) has proposed an efficient *k*-means algorithm. It does not require the number of clusters to be generated. In this algorithm, a cluster membership function is defined which assigns data objects to the nearest centroid. The proposed algorithm consists of two phases: the first phase involves pre-processing procedures that carry out the initial clustering and the second phase adjusts the centroids in a way that minimizes the newly defined membership function. Lai and Liaw (2008) improved the approach defined by Kanungo *et al.* (2002). In this, *k-d* tree is used as a data structure. They classified the clusters into two groups: static and active. Proposed approach monitors the variations in the centers of clusters to speed up the clustering process. For each node of the tree, candidates are enumerated from the active group only.

Quality of clustering algorithms decreases when the dataset has spherical shaped cluster with large variance. A solution to this problem has been provided by Fahim *et al.* (2008a). In this solution, the centroid of the large clusters is shifted towards the small clusters. For this purpose, initially, radius of the large and its neighboring small clusters is determined. If the radius of large cluster overlaps with the radius of small cluster, the average mean value of their centroids is calculated. The data objects of small clusters are grouped again using *k*-means algorithm with two initial centroids. These initial centroids are: calculated average mean value and centroid of small cluster. The data objects close to the average mean value becomes the members of large cluster and the data objects close to the centroid of small cluster does not change their membership.

Lai *et al.* (2009) measured the displacement of cluster centers during iterations. If the center of cluster does not move, *i.e.*, displacement is zero, then it means that no new member is added in the cluster; otherwise cluster is active. In their method, computation time increases linearly when using *k-d* tree, otherwise, it increases exponentially. Cluster membership and geometrical information of the data objects is considered by Lai and Huang (2010). In addition to this, a set of inequalities are also incorporated to determine the centroid of first cluster. Multiple centroid selection method is employed to determine centroids of other clusters. Jain (2010) has provided a survey of last 50 years on the popularity of *k*-means algorithm and has elaborated some of the emerging and useful research directions. Cardot *et al.* (2012) proposed a recursive algorithm for clustering large datasets. This algorithm can easily handle large sample of high dimensional data.

Bagirov *et al.* (2011) proposed an incremental approach in which a cluster center is added one at a time during iterations. This incremental approach considerably improves the standard *k*-means algorithm. However, a distance matrix is generated during iterations. The efforts of generating distance matrix get reduced by using the information from the previous iterations. Hatamlou *et al.* (2012) used gravitational search algorithm in conjunction with the standard *k*-means algorithm. It offers an enhanced *k*-means algorithm that performs better than the standard *k*-means algorithm. Nguyen *et al.* (2012) proposed a multi-view data based similarity measure. In this measure similarity between documents d_i and d_j is determined by cosine similarity measure and also considered the direction and distance of objects. Now, it is possible to use more than one object of reference in order to procure a more accurate assessment of how distant or close a pair of data objects is.

Chen *et al.* (2013) proposed a *T*wo level variable *W*eighting (*TW*) *k*-means algorithm to cluster multi-view data. Two types of weights: view weight and variable weight are used. View weight is used to identify the compactness and the variable weight is used to identify the centroid of a given cluster. Balcan *et al.* (2013) proposed two algorithms large *k*-medians and large *k*-means. Both algorithms give near optimal solution instead of an optimal solution. Near optimal solution is always faster than searching an optimum solution. Melnykov and Melnykov (2014) proposed *k*-means algorithm based on Mahalanobis instead of Euclidean distance measure. In this, initial estimation of covariance matrices is a complicated task. An initialization procedure aims to gather information for covariance matrices by identifying a group of data objects. This group of data objects are selected as centroid of clusters. Thus, they provide the estimation for covariance matrix. Membership of data objects is updated on the basis of probability. For unaffected data objects Mahalanobis distance is computed. The performance of entire strategy depends on the quality of covariance matrix.

Recently, Scitovski and Sabo (2014) proposed a technique to resolve the case when data object lies on the border of two or more clusters. In this technique, a unit weight is associated to all the data objects except the overlapped data objects. The weight of overlapped data object is uniformly divided in two or more clusters. For each participating cluster, the centroid and the objective function are calculated. The overlapped data objects become member of the cluster which gives better clustering with lower value of the objective function.

2.3.1.5 *k*-medoid and its variants

In the *k*-medoid algorithm, also called *P*artioning *A*round *M*edoid (*PAM*), medoid is the most centrally located data object in a cluster. It was proposed by Kaufman and Rousseeuw (1990). A medoid is a representative of a cluster with smallest average dissimilarity to all other data objects. This algorithm follows the same sequence of steps that are followed by the *k*-means algorithm, but the use of medoids makes it more robust to outliers. It can be used for numerical, categorical and other types of data.

Ng and Han (1994) proposed *C*lustering *L*arge *A*pplications based upon *RAN*domized *S*earch (*CLARANS*) algorithm for spatial datasets. In this algorithm, a graph of all possible *k*-medoids is formed. Every node is a one of the instance of possible *k*-medoids. Two nodes are connected if they differ by one medoid. For each node, dissimilarity value of objects with respect to their medoid is calculated and the averaged dissimilarity value is assigned as the cost of node. Then, a node is selected randomly and its cost is compared with the cost of neighboring nodes. A node with minimum cost replaces the selected node. This search process repeats until a better node is available. Eventually, a best node is returned representing *k*-medoids. Ng and Han (2002) extended their work proposed in 1994. In their extended work, *CLARANS* algorithm has been applied on polygon objects. Park and Jun (2009) proposed a variant of *k*-medoids clustering algorithm. They proposed a new initial medoids selection method. This algorithm requires calculation of distance between each pair of data objects. For each data object, an adjusted *Rand* index is calculated with the help of these distances and sorted in ascending order to select the first *k* data objects as initial medoids. Both Iris and artificial datasets are used to assess the algorithm and it is found to be computationally efficient and scalable than the *k*-medoid algorithm.

2.3.1.6 Single pass clustering

Salton (1971) introduced single pass clustering algorithm to cluster the data objects. First data object is randomly selected and assigned to the first cluster; second data object is again selected randomly and it is compared with the centroid of first cluster. If it is similar to first cluster then it is assigned as member of first cluster; otherwise forms a new cluster. The level of similarity is defined by the given threshold value. Subsequently, other data objects are clustered. Salton and Wong developed a retrieval system for this algorithm in 1978.

Arya *et al.* (1994) proposed an optimal algorithm for searching an approximate nearest neighbor. An approximate nearest neighbor data object lies within given distance from the true nearest neighbor. Hastie and Tibshirani (1996) proposed an algorithm to identify nearest neighbor in high dimensional datasets. Beyer *et al.* (1998) observed that the nearest neighbor data objects behave like the farthest neighbor data objects when dimensionality is increased. A framework applicable to all iterative clustering algorithms was developed by Bradley *et al.* (1998b). In this framework, one scan is sufficient to identify which part of data can be discarded as non-informative. Remaining part is kept in memory for analysis. This framework outperforms sampling approaches and can be extended to multiple clustering models. Aslam *et al.* (1998) proposed the Star clustering algorithm. In this algorithm, initially, the similarity between each pair is computed. Degree of each data object is determined in respect to neighbors whose distance is greater than the given threshold value. The data object with highest degree is selected as a centroid and its neighbors as the members of a cluster. For the remaining data objects, this process is repeated till other clusters are generated.

Gil-Garcia *et al.* (2003) investigated that Star clustering algorithm has two drawbacks: first, it depends on the order of selection of data object and second, it may yield illogical clusters. A modified version of Star algorithm was proposed to remove these drawbacks. Hasan *et al.* (2009) proposed an algorithm SimClus for the lower bound on similarity value. It is a greedy solution. The main difference between Star and SimClus algorithm lies in the fact that Star algorithm selects the highest degree data object as opposed to the SimClus algorithm that selects the data object covering the large number of data objects.

2.4 Hierarchical Clustering

El-Hamdouchi and Willett (1989) compared single linkage, complete linkage, group average and Ward hierarchic agglomerative clustering methods for seven documents in their test experiments and they observed that group average method is the most suitable for document clustering purposes. Brin (1995) introduced a data structure named as *G*eometric *N*ear-neighbor *A*ccess (*GNA*-tree) to store the large metric space. Handling data in a large metric space possesses its own challenges. It is, thus, based on the philosophy that the data structure should act as a hierarchical geometrical model of the data such that it reflects the intrinsic

geometry of the underlying data as opposed to simple decomposition of the data which does not use its intrinsic geometry. Zhang *et al.* (1996) proposed *B*alanced *I*terative *R*educing and *C*lustering using *H*ierarchies (*BIRCH*) algorithm. It performs a pre-clustering step by scanning the entire dataset and identified the dense regions. Dense regions are stored in a new data structure called *C*luster *F*eature tree (*CF*-tree). Now, these dense regions are clustered with the help of hierarchical clustering algorithms. Majority of hierarchical clustering algorithms perform efficiently because the dataset is divided into smaller dense regions.

Guha *et al.* (1998) proposed *C*lustering *U*sing *RE*presentatives (*CURE*) algorithm that clusters large datasets employing random sampling. It identifies non-spherical clusters and it is not sensitive to outliers. The *k*-means algorithm and its variants work well for continuous attributes. But, in the case of categorical attributes, these are converted into continuous attributes by assigning dummy values. Under such a situation, Euclidean distance is a poor measure of similarity. This is illustrated by Guha *et al.* (1999) and proposed *Ro*bust *C*lustering using lin*K*s (*ROCK*) algorithm. It is a hierarchical clustering algorithm that can measure the similarity between data objects having categorical attributes. Karypis *et al.* (1999) proposed a dynamic agglomerative clustering algorithm. It is a two steps based graph partitioning technique. In the first step, small dense clusters are built using graph partitioning method and in the second step, agglomerative algorithm is performed. A relative inter-connectivity and intra-connectivity similarity is measured for the clusters. This algorithm identifies clusters of different shapes, densities, and sizes in two-dimensional space. Chiu *et al.* (2001) proposed a new hierarchical clustering method. They also proposed a new distance measure for both continuous and categorical attributes. This distance measure is derived from probabilistic model which uses log-likelihood function. They used the framework of Zhang *et al.* (1996) for clustering the data objects.

2.5 Density based Clustering

Ester *et al.* (1996) proposed first density-based algorithm known as *D*ensity-*B*ased *S*patial *C*lustering of *A*pplications with *N*oise (*DBSCAN*). In this algorithm, clusters grow by including the dense neighborhoods data objects. Arbitrary shaped clusters are identified using *DBSCAN* algorithm. It requires two parameters: *eps* and *minpts*; *eps* denotes the threshold

distance value and *minpts* describes the minimum number of data objects within threshold distance (*eps*) in a cluster. They also estimated the values of both parameters. This algorithm, however, is prone to errors when clusters are not well-separated.

Hinneburg and Keim (1998) proposed a density based algorithm known as *DEN*sity *CLU*st*E*ring (*DENCLUE*) for multimedia datasets. The basic idea of this approach is to model the overall object density analytically as the sum of influence functions. Arbitrary shaped clusters are generated based on the overall density function. Ankerst *et al.* (1999) proposed an extension of *DBSCAN* algorithm named as *O*rdering *P*oints *T*o *I*dentify the *C*lustering *S*tructure (*OPTICS*). In this algorithm, the ordering of data objects is based on two distance parameters: core distance and reachable distance. These two parameters determine the density and reveal the hidden structure in the dataset. Fahim *et al.* (2008b) proposed *D*ensity *C*lustering *B*ased on *O*utlier *R*emoval (*DCBOR*) algorithm. It identifies clusters of any shape, size and density. It uses a threshold value to detect outliers. A graph of *k* nearest neighbors is formed and high density data objects are taken as initial centroids. Low density data objects are removed as outliers.

2.6 Genetic based Clustering

Minz and Jain (2005) proposed hybridized rough set framework. It is composed of traditional rough sets and classical decision tree induction algorithms. Rough sets help to identify dominant attributes and decision tree is used as a generalized classifier. Panigrahi *et al.* (2009) used genetic algorithm to detect damages in a uniform strength beam. Rahman and Verma (2013) proposed a layered clustering architecture for classifiers. Number of layers and number of clusters per layer is decided by the genetic algorithm. They optimized these two parameters. When number of cluster differs from one layer to other layer then a majority voting is used to take a proper decision.

2.7 Hypergraph based Clustering

Han *et al*. (1997) proposed an algorithm to cluster in high dimensional spaces using hypergraphs. Hypergraphs are the extension of the regular graphs. In hypergraphs, an edge joins many vertices known as hyperedge. Graph-partitioning algorithm partitions the hypergraph into two parts in such a way that the weight of the hyperedges cut is minimized. The partitioning process is continued until a fixed number of partitions are achieved. A

software package was developed by Karypis and Kumar (1998) for partitioning large hypergraphs. It produces high-quality partitions at a high rate for a variety of hypergraphs. Xiong *et al.* (2009) analyzed the effects of the data distributed in the dataset. They have illustrated that *k*-means algorithm generates uniform sized clusters whereas actual clusters may be non-uniform.

2.8 High Dimensional Clustering

High dimensional dataset means data objects with three or more attributes. Quality of clustering can be judged only up to three dimensions as it is difficult to visualize 3-D data on a computer screen. Data objects in high-dimension can be sparse and highly skewed. It is hard to build a mental image of the data. High dimensional data have become prevalent in different applications such as in pattern recognition, machine learning and data mining. The data objects in data mining could have hundreds of attributes. The definition of high dimension has also changed from tens of features to hundreds or even tens of thousands of features. Clustering in such high dimensional spaces presents tremendous difficulty. Traditional clustering algorithm find clusters in the full space of the datasets but now research has shifted to find clusters embedded in the subspaces (subset of attributes). Subspace clustering is divided into two approaches: top down and bottom up.

Agrawal *et al.* (1998) proposed the first bottom-up subspace clustering algorithm, namely CLustering *I*n *QUE*st (*CLIQUE*) for high dimensional clustering. It is a combination of grid and density based approaches. Clustering starts at single dimension and grows upwards to full dimensional space. Each dimension is partitioned into same number of equal length intervals on the basis of grid. Each unit has the same volume, and therefore the number of points inside it can be used to approximate the density of the unit. *CLIQUE* uses three steps: first it identifies the subspace, second it identifies the clusters and lastly it generates minimal description for the clusters. It uses two parameters, numbers of intervals and density threshold as input parameters. Goil *et al.* (1999) have proposed *M*erging of *A*daptive *F*inite *I*nterv*A*l (*MAFIA*) which significantly modifies *CLIQUE*. In *MAFIA* algorithm, adaptive grid has been used in contrast to *CLIQUE* which uses a fixed size grid. It starts with one data pass to construct a minimum number of variable intervals in each dimension. Contiguous intervals with similar histogram values are merged. Based upon low density bins and cells are pruned

to reduce the computation. In *MAFIA*, along with density threshold parameter, the threshold for merging adjacent cells needs to be specified by the user. On similar datasets, it is faster than *CLIQUE*, although, the running time has an exponential relation with respect to dimension of dataset.

Aggarwal *et al.* (1999) proposed the first top-down subspace clustering algorithm *PROjected CLUS*tering (*PROCLUS*). Similar to *CLARANS*, *PROCLUS* takes samples of data to cluster the dataset. The algorithm uses three phases: initialization, iteration, and cluster filtration. In initialization phase, a good superset of piercing set of medoids is selected using greedy approach from samples of data. The iteration phase selects a random set of *k* medoids from the output set of first phase. It replaces ruthless medoids with new medoids, and determines if clustering has improved by hill climbing technique. Cluster quality is based on the average distance between instances and the nearest medoid. For medoid, a set of dimensions (subspace) is chosen whose average distances are small compared to statistical expectation. Once the subspaces have been selected for each medoid, average Manhattan segmental distance is used to assign points to medoids, forming clusters. The filtration phase computes new dimensions for each medoid based on the clusters formed and reassigns points to medoids, removing outliers.

Aggarwal and Yu (2000) proposed an extended version of *PROCLUS* called *OR*iented *CLUS*tering (*ORCLUS*) that looks for arbitrarily oriented projected subspaces of lower dimensionality. This algorithm highlights the inter-attribute dependency among data objects. Overall algorithm can be divided into three steps: assign clusters, find subspace vectors corresponding to cluster, and merge. During the assign phase, the algorithm iteratively assigns data points to the nearest cluster centers. Subspace vector determination redefines the subspace associated with each cluster by calculating the covariance matrix for a cluster and selecting the orthonormal eigenvectors with the least spread (smallest eigen values). Clusters that are near each other and have similar directions of least spread are merged during the merge phase. The number of clusters and the size of the subspace dimensionality must be specified. The extended cluster feature vectors have been included in this method which makes it scalable to very large datasets. *ORCLUS* uses random sampling to improve speed and scalability and as a result may miss smaller clusters.

Milenova and Campos (2002) proposed the O-Cluster clustering algorithm for high dimensional datasets. In this novel method of clustering, an active sampling technique is combined with a partitioning strategy in order to identify the continuous areas of high density present in the input space. A limited memory buffer and at the most single scan through the data are required in this algorithm. Yang *et al.* (2002) proposed δ-clusters, which is a subspace clustering method. The coherence of a subset of data objects is captured on a subset of attributes, while allowing attribute values to be absent. Using these methods, similar trends in data objects which sometimes can be visualized only by shifting the attributes are identified. As a measure of similarity, Pearson's correlation coefficient is used to find the coherence among data objects. There are two types of coherence: shifting coherence and amplification coherence. The major emphasis is on shifting coherence in their work. Residue is introduced to measure the degradation of the coherence of the cluster. On the basis of the data object and attribute bias within the cluster, the difference between the actual and expected value of each entry is calculated. The input parameter taken by this algorithm is the number of clusters as well as the cluster size of each individual cluster. Friedman *et al.* (2002) proposed *C*lustering *O*n *S*ubsets of *A*ttributes (*COSA*), a subspace clustering algorithm for considering data where number of attributes is much larger than the number of data objects. It starts with assigning equal weight to all dimensions. The *k -N*earest *N*eighbors (*k-NN*) algorithm is used to examine each instance. Maximum weight is assigned to the attribute with smallest dispersion within each group. All other attributes are given zero weight, regardless of their dispersions. The dimension weights for pair of instances are calculated using these weights in order to update the distances that are used in the computation of *k-NN*. This process is iterated until the weights get converged to some value.

Woo *et al.* (2002) proposed a *F*ast and *IN*telligent subspace clustering algorithm using *DI*mension vo*T*ing (*FINDIT)* that works like *PROCLUS*, but none of previous algorithms far have exhibited the robust result of the noise ratio and the cluster densities. *D*imension-*O*riented *D*istance (*DOD*), a new distance measure has also been devised in this method for subspace clustering. It utilizes the dimensional and value difference information together. In addition, the number of places, where the difference in the values of components of two data objects is less than the threshold value, is counted. It has three phases: the sampling phase, the cluster forming phase, and the data assigning phase. In sampling phase, two different sets

of samples are made from the dataset depending upon desired size of cluster. These sets determine the initial representative medoids of the clusters. Clustering formation phase is iterated several times while increasing input parameter given to measure the difference in value of two points. Concept of dimensional voting is used to measure the quality of clusters. As a part of the final phase, all the data points are assigned to one or other medoids based upon the subspace found in the second phase, *i.e*., cluster forming phase.

A space-partitioning technique, called as *C*ell *B*ased *F*iltering (*CBF*) was proposed by Chang and Jin (2002). In addition, it also incorporated a filtering-based index structure by using an approximation technique. It focuses on efficient use of main memory both for high dimension and large datasets. This method provides cell creation and insertion algorithms. In cell creation algorithms, by splitting each dimension into a set of partitions, cells are created that use a split index. First, the optimal split section is found by repeatedly examining a value between the maximum and the minimum in each dimension. It creates a cell from *n* split sections for *n*-dimensional data that makes much less cells than in *CLIQUE*. The cell insertion algorithm constructs a cluster as a cell with more density than a given threshold. Now, it stores the constructed cluster into an index structure. *D*ensity-based *O*ptimal projective *C*lustering (*DOC*) is an algorithm proposed by Procopiuc *et al*. (2002). In this, the projective clusters are computed iteratively and the points closed in subspace are grouped together. Each cluster is projected on its associated subspace, as opposed to the entire dataset on one subspace. Once an optimal projective cluster is identified, the algorithm is iteratively run in a greedy manner till the termination condition is met. During the iteration, an approximation of an optimal cluster over the current set of points is computed. The termination condition could either be the percentage of the points clustered, quality measures of the clusters or the user specified number of clusters. This algorithm never misses the small clusters and that is one of its biggest advantages. Kailing *et al.* (2004) discussed density-connected subspace clustering for high dimensional databases. Parsons *et al.* (2004) has given extensive survey on subspace clustering for high dimensional data.

Pilevar and Sukumar (2005) also presented a grid clustering algorithm namely *G*rid *C*lustering algorithm for *H*igh-dimensional very *L*arge spatial datasets (*GCHL*) for spatial datasets. Patrikaninen and Meila (2006) compared the subspace clustering methods. A

dimensionality unbiased subspace clustering is proposed by Assent *et al.* (2007). A novel distance function for subspace clustering and two visualization techniques are proposed. Zaki *et al.* (2007) presented an efficient algorithm that operated on categorical datasets for mining subspace clusters. Their algorithm is known as *S*ubspace *CL*uster*I*ng of *C*ategorical data via maximal *k*-partite cliques (*CLICKS*). Chu *et al.* (2010) proposed *DEN*sity *Co*nscious *S*ubspace (*DENCOS*) clustering algorithm which is a variant of density based clustering algorithm for solving the density divergence problem. This problem states that region densities vary in different subspace cardinalities. In all the subspaces, according to their work, the previously done work took only fixed density threshold as a tool to discover the dense regions. As a consequence, clustering accuracy was lost in different subspace cardinalities. In response to this problem, a novel subspace clustering model has been proposed. In this model, the clusters are regions with densities higher than region densities in a subspace. The clusters are then computed based on the relative region densities. As an extension of this, to determine the clusters in different subspace cardinalities, density thresholds are adaptively determined. *DEN*sity *Co*nscious *S*ubspace (*DENCOS*) algorithm is a clustering technique which uses a divide-and-conquer scheme. Its objective is to efficiently discover clusters that satisfy various density thresholds in different subspace cardinalities.

2.8.1 Dimensionality reduction

The curse of dimensionality exists in the high dimension datasets. To solve this problem, there is a need to reduce the dimensions of the dataset. Entropy measurements are the basis of the filter model approach presented by Dash *et al.* (1997). The authors proved that entropy tends to be low for data that contains tight clusters. Thus, entropy is a good measure for determining the feature subset relevance.

An often important preprocessing step is feature transformation. It allows the clustering algorithm to utilize a few of the newly created features. As an improvement over the existing methods for clustering by Hinneburg and Keim (1999), clustering methods use such feature transformations. This aids in identifying the important features in addition to iteratively improving clustering. Murtagh *et al.* (2000) used wavelet transforms to reduce the dimensions of a dataset. They used canonical ordering of observations and variables, applied a wavelet transform, modeled the noise and defined significant parts and then read off the

resultant clusters. Donoho (2000) argued that high dimensionality in data analysis is both a curse and a blessing. Most of the mathematicians discussed the curse of dimensionality widely but a few pondered over the blessing. In this paper, both the domains of curse and blessing have been explored.

Dhillon and Modha (2001) introduced the concept of decomposition for the documents having high dimensionality and sparsity. They approximated the errors of the concept decompositions close to the truncated singular value decompositions. Achtert *et al.* (2007) proposed *D*etect*I*ng *S*ubspace cluster *H*ierarchies (*DISH*) algorithm that detects clusters of different sizes, shapes, and densities in subspaces of different dimensions. It also uncovers complex hierarchies of nested subspace clusters. Boutsidis and Magdon-Ismail (2013) proposed the first deterministic feature selection algorithm for *k*-means. This method is used for decomposition of the identity. Their approach also includes a structural result which quantifies some of the trade offs in dimensionality reduction. Feature selection for *k*-means algorithm has also been proposed by Mavroeidis and Marchiori (2014).

2.9 Applications of Clustering

Freund and Schapire (1997) allocate the resources dynamically in the on-line framework. The model studied by them was an abstract and a broader extension of the previously well studied clustering model in the context of a general and decision-theoretic setting. Rose (1998) provided deterministic annealing for clustering, compression, classification and regression related optimization problems. Brin and Page (1998) discussed a method to build such systems which can exploit the additional information present on the web. They also discussed how to deal with the uncontrolled documents on web where anyone is permitted to publish anything. They have given reasons why their *Google* search engine is more efficient than other search engines.

Boley *et al.* (2000) categorized the web documents using a partition based clustering. Since, web documents have high dimension as compared to the size of documents, authors proposed a graph partitioning methodology which does not require the parameters like similarity measure. Steinbach *et al.* (2000) compared document clustering algorithms such as the standard and bisecting *k*-means along with hierarchical clustering. They found that the

bisecting *k*-means algorithm was better than the standard *k*-means and hierarchical clustering. Karypis and Han (2000) proposed concept indexing for the document retrieval and categorization in high dimensional spaces.

Ertoz *et al.* (2001) not only produced general clusters but also coherent clusters for document. They emphasized that while documents are being clustered, membership of one document may be more similar to and better suited for some document belonging to the other cluster but their major theme does not allow them to be in same cluster. Merz (2003) proposed an iterative local search technique. Initially, local search is applied to randomly available solution. Then, local optimum is mutated and local search is again applied on mutated value in order to search next local optima. New solution is accepted only if it gives better objective function value. This procedure is repeated for fixed number of times.

Toshniwal and Joshi (2005) proposed cumulative weighted slopes for clustering time series. They observed that similar time sequences have similar slopes at their corresponding points. The weighted sum of these slopes is called cumulative weight slope and is computed for each time sequence. Clusters are formed on the basis of this weighted sum of slopes to identify similar patterns over periods of time. Banerjee and Ghosh (2006) discussed the scalability of clustering algorithm with balancing constraints. Beg and Ahmad (2007) proposed seven user feedback parameters to produce better query results in response to the query posted. Some of the fuzzy rank ordering techniques and a few novel techniques are used to rank the documents that are preferred by the individuals. This is helpful in creating personalized web search engine. Damian *et al.* (2007) discussed the applications of subspace clustering algorithm *COSA* in medical system biology.

Ashraf *et al.* (2008) focused on automatic information extraction from *HTML* documents. The data from an *HTML* document is parsed and tokenized followed by partitioning it into clusters that contain similar elements. An extraction rule is then estimated based on the pattern of occurrence of data tokens. A system named as *CluTex* is proposed by them. HTML document extraction is a classification task but in this work, they have solved it as a clustering task. Harper *et al.* (2013) examined the price volatility in the silver spot market. For the analysis and better understanding of the volatility, a host of *G*eneralized *A*uto *R*egressive *C*onditional *H*eteroskedasticity (*GARCH*) models are used. It was found that the

GARCH model indicates a high significance of the spot price of silver in the past and it also suggests that over time, volatility does not remain constant. Saabni *et al.* (2014) proposed an automatic text line extractor from historic documents without any barrier to its language. It works for both binary and gray scale images. They also created a dataset of historic documents providing various challenges to the researchers.

2.10 Validation Measures

Davies and Bouldin (1979) proposed a validity measure, namely, *D*avies *B*ouldin (*DB*) index, to measure the quality of clustering. It measures the compactness of data objects in a cluster and the separation between them. The *DB* index proposed by them is the averaged ratio of compactness of two clusters to the separation between them. This index is independent of the number of cluster and clustering technique involved.

Rousseeuw (1987) proposed a method of displaying cluster's data objects graphically. Two things are required for this: cluster members and proximity between the data objects. An index is measured for each data object which is the ratio of difference between averaged distance of data objects in nearest cluster and averaged distance of data objects in same cluster to the maximum of both averaged distance. This values lies between -1 and 1. This value can be used to display how tightly a data object is bound to a cluster.

Halkidi *et al.* (2000) emphasized on evaluation of clusters by measuring their quality. Quality of clustering algorithm depends on degree to which it fits a specific data set and the optimal number of clusters present in the dataset. Halkidi *et al.* (2001) reviewed the fundamental concepts of clustering algorithms. They also addressed the issue of assessing the quality of clustering. They emphasized that quality of clustering depends on the inherent property of data under consideration. Halkidi *et al.* (2002a) presented a survey on various cluster validation techniques. In this survey, both internal and external validity measures have been explored. Halkidi *et al.* (2002b) presented another survey on various clustering validation techniques. This survey is based on the relative validity measures.

Chapter Summary

In this chapter, a survey on data mining, determination of the number of clusters, centroids initialization methods, outlier detection methods, variants of *k*-means, *k*-medoid and single pass clustering has been carried out. Literature on hierarchical, density, grid, genetic, hypergraph and high dimensional clustering has also been reviewed. Related applications and validation measures of clustering have also been presented in this chapter.

Chapter 3

Modified Single Pass Clustering Algorithm

Primary goal of clustering methods is to find the interesting patterns in a given dataset. Partitioning based methods, one of the popular clustering methods, is the major focus of this research work. There are many algorithms like *k*-means, *k*-medoid, *k*-mode and single pass clustering available in literature under the partitioning based methods. This chapter describes the detailed description of *k*-means and single pass clustering algorithms. A modified single pass clustering algorithm has also been proposed in this chapter. The proposed algorithm is compared with *k*-means and single pass clustering algorithms on the artificial and real datasets.

3.1 Introduction

In the partitioning based methods, *n* data objects are partitioned into *k* clusters based upon the user defined parameters like number of clusters or a threshold similarity value. Each cluster is represented by a centroid which is an arithmetic mean of all the data objects of that cluster. Data objects are partitioned into clusters that optimize a clustering criterion. One of the criteria is to keep the minimum sum of squared distance between data objects and the centroids of the clusters. Requirement of the user's prior knowledge about the number of clusters or threshold similarity value motivates the researchers to find an efficient clustering algorithm. A modified single pass clustering algorithm has been proposed which does not require the user defined parameters. It automatically determines the threshold similarity value.

3.2 Types of Partitioning based Methods

There are two types of partitioning based methods: single pass clustering and multi-pass clustering. Single pass clustering, as the name implies, requires only one pass to cluster the given data objects; the time complexity of this method is of the order $O(nk)$. Multi-pass clustering is an iterative approach to cluster the given dataset; the time complexity of this method is of the order $O(nkl)$. Here, n is the number of data object, k is the number of clusters and l is the number of iterations.

3.2.1 *k*-means algorithm

k-means is a popular multi-pass partitioning based algorithm. It aims to find *k* number of clusters that have minimum sum of squared distance between the data objects and the centroids. In this algorithm, *k* is a user defined parameter and centroid is the mean value of data objects of a cluster. Distance between data objects $\boldsymbol{d_i}$ and $\boldsymbol{d_j}$ is calculated using Euclidean distance measure which is defined as:

$$dis(\boldsymbol{d_i},\ \boldsymbol{d_j}) = \sqrt{(x_i^1 - x_j^1)^2 + (x_i^2 - x_j^2)^2 + \ \dots\ + (x_i^d - x_j^d)^2} \quad \dots (1)$$

where, $\boldsymbol{d_i} = (x_i^1,\ x_i^2, \dots ,\ x_i^d)$ and $\boldsymbol{d_j} = (x_j^1,\ x_j^2, \dots ,\ x_j^d)$ are *d*-dimensional data objects. Let $D = \{d_1,\ d_2, \dots ,\ d_n\}$ be the dataset of *d*-dimensional data objects that need to be partitioned into a set of *k* clusters ($K = \{k_1,\ k_2, \dots ,\ k_k\}$). Each cluster has its own representative called centroid which is also a *d*-dimensional object. Centroids are represented by a centroid set $C = \{c_1,\ c_2, \dots\ ,\ c_k\}$. Centroid $\boldsymbol{c_j}$ is a *d*-dimensional object of cluster k_j, the value of i^{th} dimension of centroid $\boldsymbol{c_j}$ is calculated as:

$$c_j^i = \frac{1}{m}\sum_{p=1}^{m} x_p^i \quad \dots (2)$$

where, *m* is the number of data objects in cluster k_j and x_p^i is the value of i^{th} dimension of p^{th} data object belonging to cluster k_j. Similarly, other dimensions of centroid $\boldsymbol{c_j}$ are calculated. Initially, data objects are not the members of any cluster; *k* centroids are selected from the given dataset as the initial centroids of the *k* clusters; the distance of each data object from the *k* centroids is calculated using (1); data object is assigned as the member of cluster when the distance of this object is least from the centroid of that cluster. Now, centroid of each cluster is updated using (2). This process is repeated until a data object changes its cluster. This algorithm is simple to implement, so it is widely used for clustering. These steps of *k*-means algorithm are depicted in Figure 3.1.

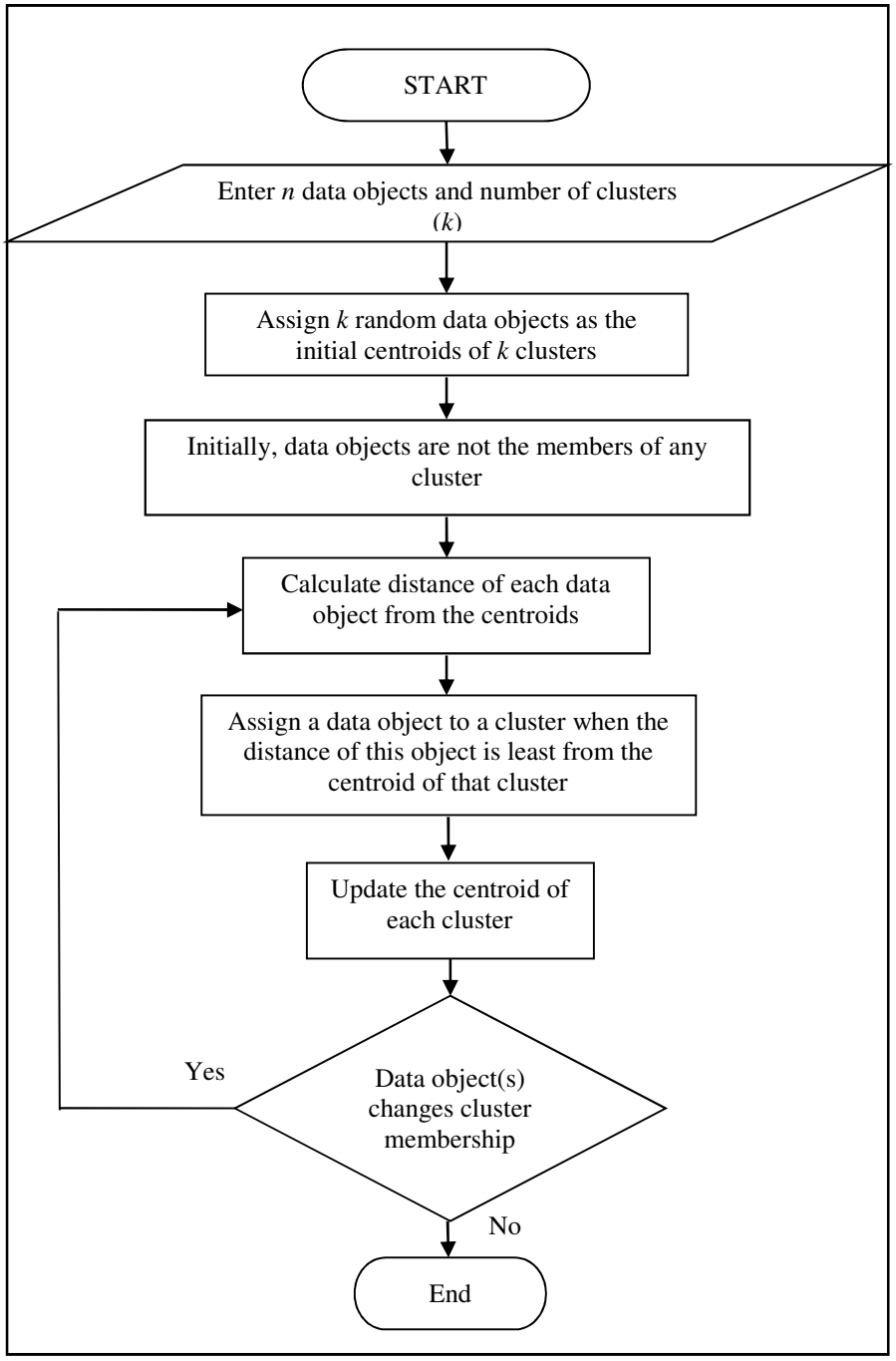


Figure 3.1: Flow chart of *k*-means algorithm

3.2.2 Single pass clustering algorithm

*S*ingle *P*ass *C*lustering (*SPC*) is a popular one pass partitioning based algorithm. This algorithm is used to cluster the data objects based on the user defined threshold similarity value (T_{th}). Threshold similarity value is the maximum permissible distance between the data

object and the centroid of a cluster. Initially, a random data object is assigned to the first cluster; this data object also serves as the initial centroid. Remaining data objects are iteratively merged into the existing clusters or form a new cluster based upon the given threshold similarity value. Unlike *k*-means algorithm, all the data objects need not to be in main memory during iterations; but like *k*-means algorithm, centroids are always present in the main memory. The steps involved in *SPC* algorithm are depicted in Figure 3.2.

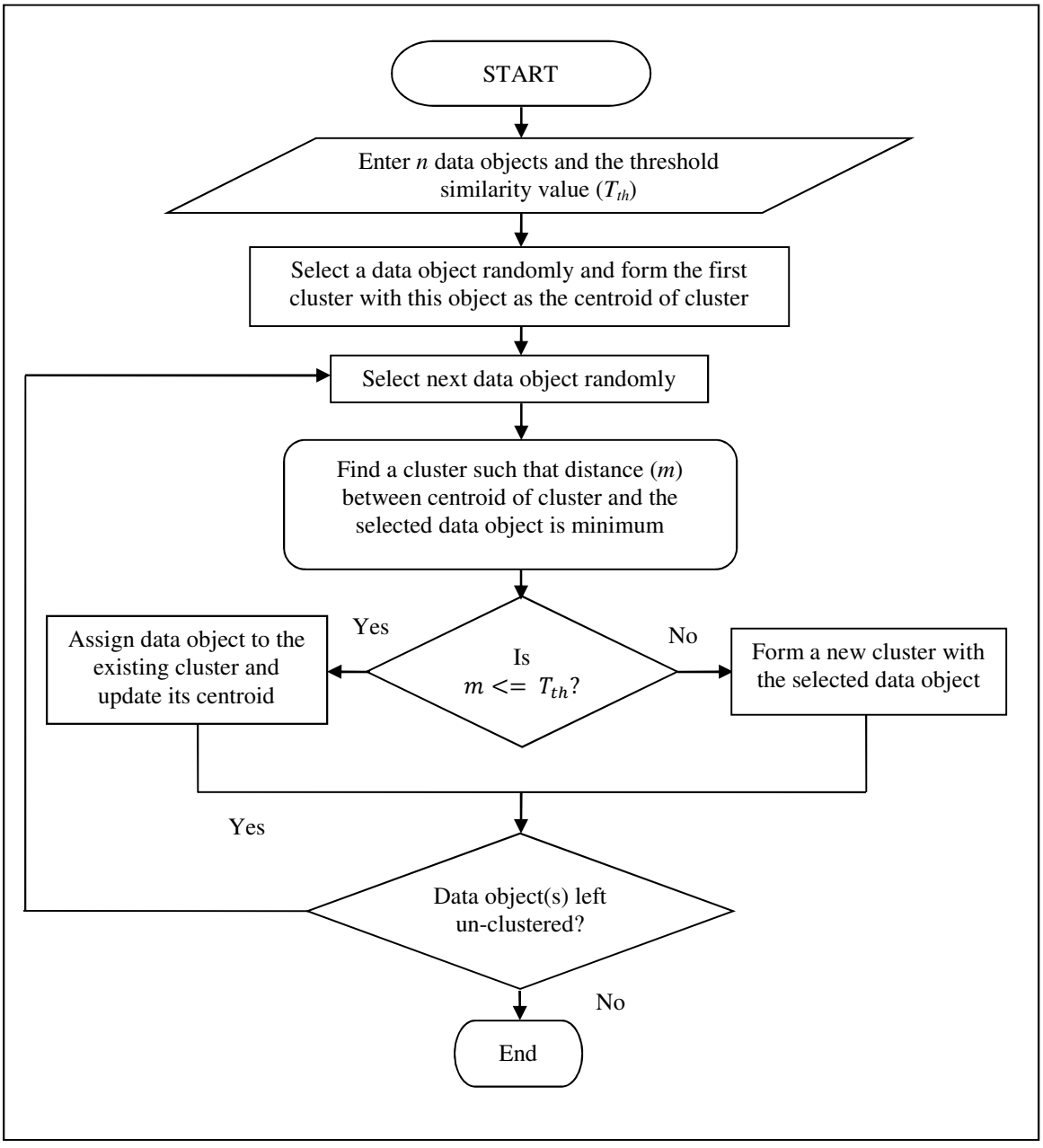


Figure 3.2: Flow chart of *SPC* algorithm

3.3 Limitations of *k*-means and *SPC* Algorithms

A critical look at the *k*-means and *SPC* algorithms uncover the following shortcomings:

- In *k*-means clustering algorithm, the number of clusters (*k*) needs to be specified beforehand.
- *k*-means algorithm is dependent on the selection of initial centroids. Due to this, *k*-means is vulnerable to the local optima and may miss the global optima. Consequently, this suboptimal clustering requires multiple runs with updated initial centroids.
- *k*-means algorithm is sensitive to outliers. Their presence can affect the quality of clustering.
- *k*-means algorithm initializes cluster centroids randomly and iteratively updates them to have minimum sum of squared distance between the data objects and the centroids of clusters. Nevertheless, iterations are indefinite as *k*-means algorithm terminates only when stable clusters are formed.
- *SPC* algorithm depends on the order of selection of data objects.
- *SPC* algorithm requires threshold similarity value as a user defined parameter.
- *k*-means and *SPC* algorithms generate spherical clusters.

To overcome these limitations, a modified *SPC* algorithm has been proposed in this work.

3.4 Modified Single Pass Clustering Algorithm

*M*odified *S*ingle *P*ass *C*lustering (*MSPC*) algorithm revolves around the proposition of a threshold similarity value. This is not a user defined parameter; instead, it is a function of data objects left to be clustered. In this algorithm, data objects are selected randomly and assigned to either one of the existing clusters or form a new cluster based on the threshold similarity value. Here, threshold similarity value is not a constant; rather, it is updated during the execution of proposed algorithm. In our experiments, this threshold similarity value is taken as mean/median of the paired distance of all data objects left to be clustered. An adjacency matrix (A) is used to store the paired distance between all the data objects. Once a data object is clustered, its distance to other data objects is removed from the adjacency matrix; and threshold similarity value is updated to a new value.

Threshold similarity value as a mean is calculated as:

$$T_{th}(= f(A)) = \frac{2}{t(t-1)} \sum_{i=1, j=i+1}^{n} a_{ij} \quad \ldots (3)$$

The threshold similarity value as a median is calculated when a_{ij}'s are sorted:

$$T_{th}(= f(A)) = \left\{ Value\ of\ a_{ij}\ at\ postition\ \left(\frac{t(t-1)}{4}\right)\ when\ a_{ij}' \text{s are sorted} \right\} \quad \ldots (4)$$

where, $A\ (= a_{ij})$ is the adjacency matrix, a_{ij} being the distance between data objects $\boldsymbol{d_i}$ and $\boldsymbol{d_j}$ and *t* is the number of data objects left to be clustered.

The proposed algorithm consists of the following steps:

a) Select a data object randomly and assign it as the member and the centroid of first cluster.
b) Threshold similarity value (T_{th}) is calculated as a function of data objects left to be clustered.
c) Select the next data object randomly. This data object will belong to either one of the existing clusters or will form a new cluster. Selected data object is assigned to a cluster with the help of distance between the data object and centroids of existing clusters; and the threshold similarity value. If the distance between the selected data object and the centroids is more than the threshold similarity value, a new cluster is formed and selected data object becomes the member and the centroid of new cluster; otherwise, it is assigned to one of the existing clusters whose centroid has minimum distance from the selected data object.
d) Update the centroid of existing cluster when the selected data object is added in the cluster.
e) Update threshold similarity value (T_{th}) as a function of data objects left to be clustered.
f) Repeat steps c), d) and e) until all the data objects have been clustered.

These steps are presented below in algorithm *MSPC*(*D*, *A*).

Algorithm: MSPC (D, A)

//Input: Let $D = \{d_1, d_2, \dots, d_n\}$ be a set of *n* data objects to cluster and a set
$A=\{a_{ij} | a_{ij} = distance\ between\ data\ objects\ d_i\ and\ d_j\ for\ 1 \le i,j \le n\ and\ j > i\}$

//Output: A set $K = \{k_1, k_2, \dots, k_k\}$ denotes *k* clusters and a set $C = \{c_1, c_2, \dots, c_k\}$ denotes centroids of these clusters.

- $m = 1$;
- $t = n - 1$;
- $k_m = \{d_p | \exists d_p \in D\}$; $//randomly\ select\ a\ data\ object\ d_p from\ D$
- $K = \{k_m\}$;
- $c_m = d_p$;
- $C = \{c_m\}$;
- $\boldsymbol{for}$ $each\ unclustered\ data\ object\ d_{q'}$
- $\boldsymbol{do\ if}$ $d_{q'}\ is\ chronologically\ greater\ than\ d_p$
- $\boldsymbol{then}$ $A = A - \{a_{pq'}\}$;
- $\boldsymbol{else}$ $A = A - \{a_{q'p}\}$;
- $T_{th} = f(A)$;
- $\boldsymbol{for}$ $each\ random\ data\ object\ d_q \in \{D\} - d_p$
- $\boldsymbol{do\ for}$ $each\ centroid\ c_r \in C$
- $\boldsymbol{do}$ $s_{c_r} = dis(d_q, c_r)$; $//dis(d_q, c_r)\ is\ the\ distance\ between\ d_q\ and\ c_r$
- $s_{c_j} = min(s_{c_1}, s_{c_2}, s_{c_3}, \dots, s_{c_r})$;
- $\boldsymbol{if}$ $(s_{c_j} \le T_{th})$
- $\boldsymbol{then}$ $k_j = k_j \cup d_q$;
- $Update\ centroid\ c_j\ of\ cluster\ k_j$;
- $\boldsymbol{else}$ $m = m + 1$;
- $k_m = \{d_q\}$;
- $K = K \cup \{k_m\}$;
- $c_m = d_q$;
- $t = t - 1$;
- $\boldsymbol{for}$ $each\ data\ object\ d_{q'} \in \{D\}\ not\ clustered$
- $\boldsymbol{do\ if}$ $d_{q'}\ is\ chronologically\ greater\ than\ d_q$
- $\boldsymbol{then}$ $A = A - \{a_{qq'}\}$;
- $\boldsymbol{else}$ $A = A - \{a_{q'q}\}$;
- $T_{th} = f(A)$;

3.5 Performance Evaluation of *MSPC* Algorithm

In *k*-means algorithm, *k* random data objects are considered as the initial centroids; in the *SPC* algorithm, a constant threshold similarity value is given by the user; and in the proposed

MSPC algorithm, mean or median of paired distance of data objects has been considered as threshold similarity value. This section is divided into two sub-sections; in the first sub-section, the experiments have been performed on the artificial datasets; in the second sub-section, the experiments have been performed on the real datasets.

3.5.1 Artificial datasets experiments

To perform the experiments, fifteen artificial datasets each containing five hundred *2-D* data objects have been created. These data objects are generated randomly in the range of 100 to 499 in both dimensions. In *k*-means algorithm, the value of *k* is taken in the range of 4 to 10, in *SPC* algorithm, the value of threshold is taken in the range of 50 to 75 and in *MSPC* algorithm, threshold similarity value is calculated using mean and median, using (3) and (4) respectively. In these experiments, quality of clustering algorithms is assessed by means of separation and compactness validity measures and further, they are validated on Dunn and *DB* validity indices.

3.5.1.1 Experiments using mean as threshold similarity value

Figures 3.3 - 3.5 show the single linkage, complete linkage and centroid linkage separation measures, respectively, for *k*-means, *SPC* and *MSPC* algorithms.

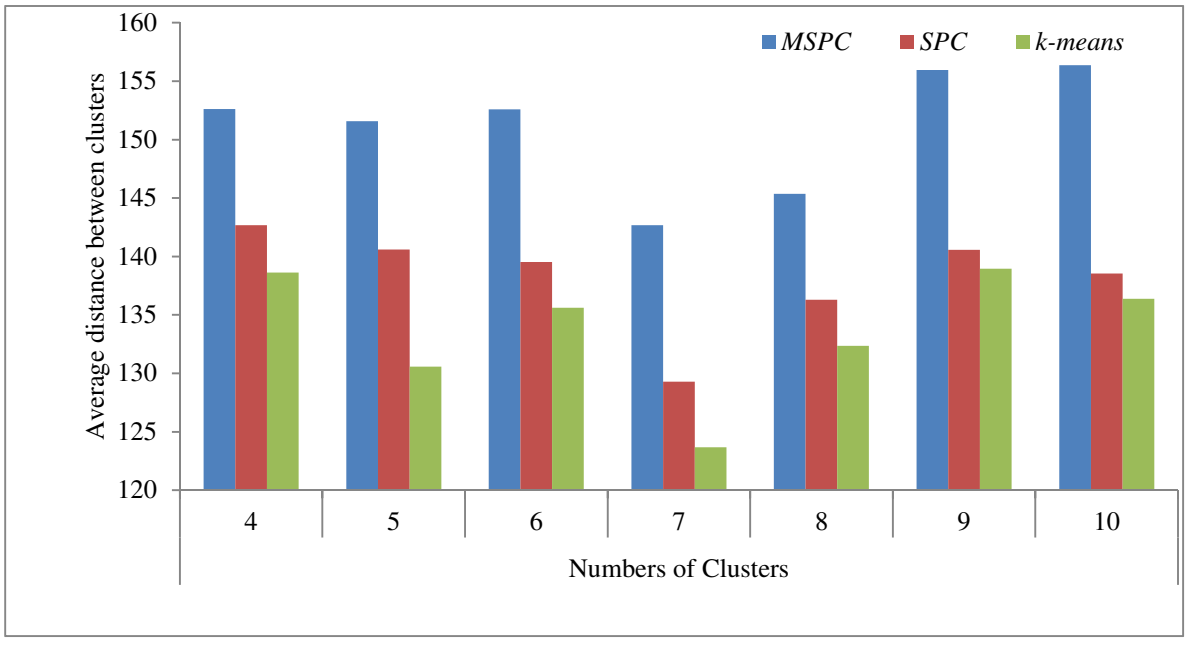


Figure 3.3: Single linkage separation using mean value as a threshold

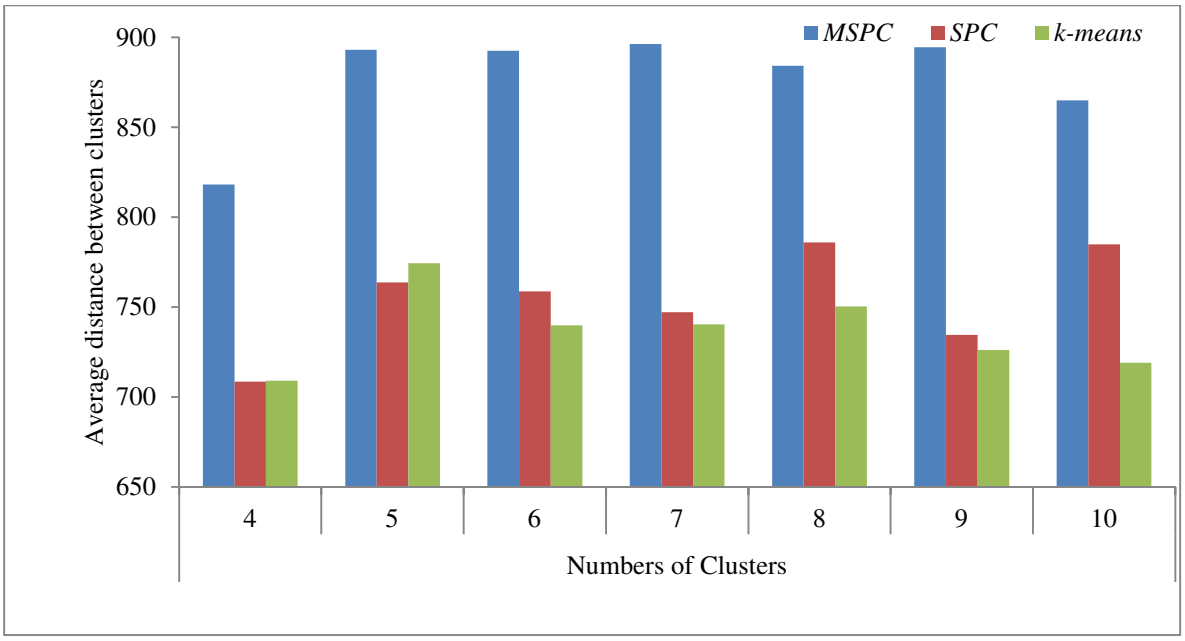


Figure 3.4: Complete linkage separation using mean value as a threshold

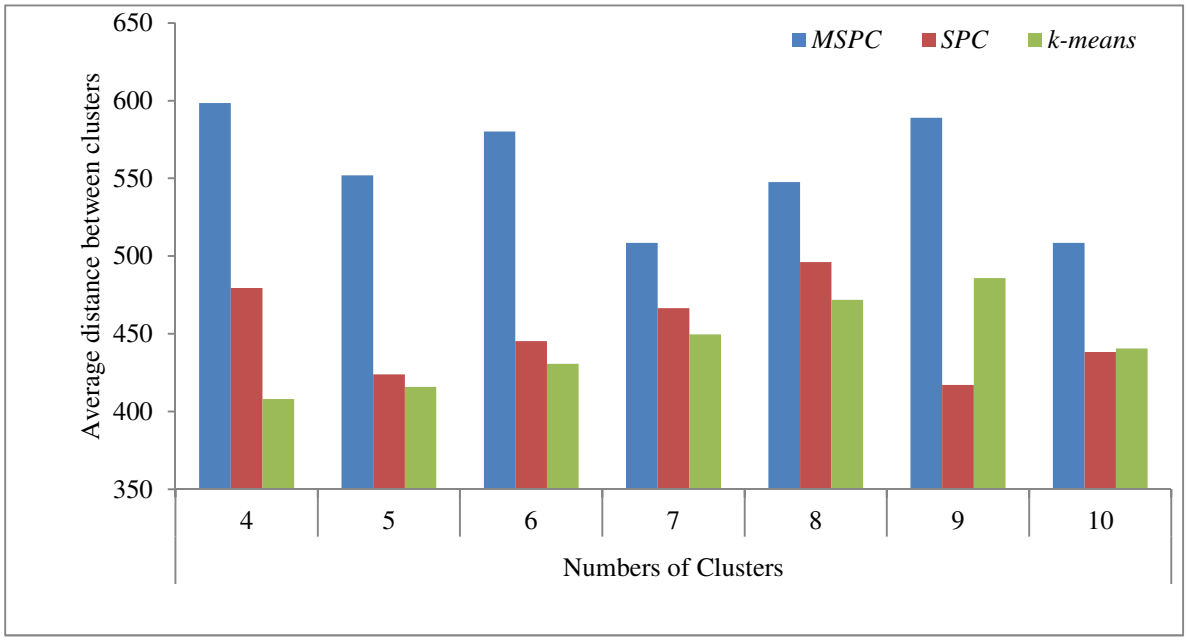


Figure 3.5: Centroid linkage separation using mean value as a threshold

From these figures, it can be observed that clusters generated by *MSPC* algorithm are more separated than the clusters generated by *k*-means and *SPC* algorithms.

Figures 3.6 and 3.7 show the centroid linkage and averaged paired compactness measures, respectively, for *k*-means, *SPC* and *MSPC* algorithms. It can be observed from these figures

that clusters generated by *MSPC* algorithm are more compact than the clusters generated by *k*-means and *SPC* algorithms.

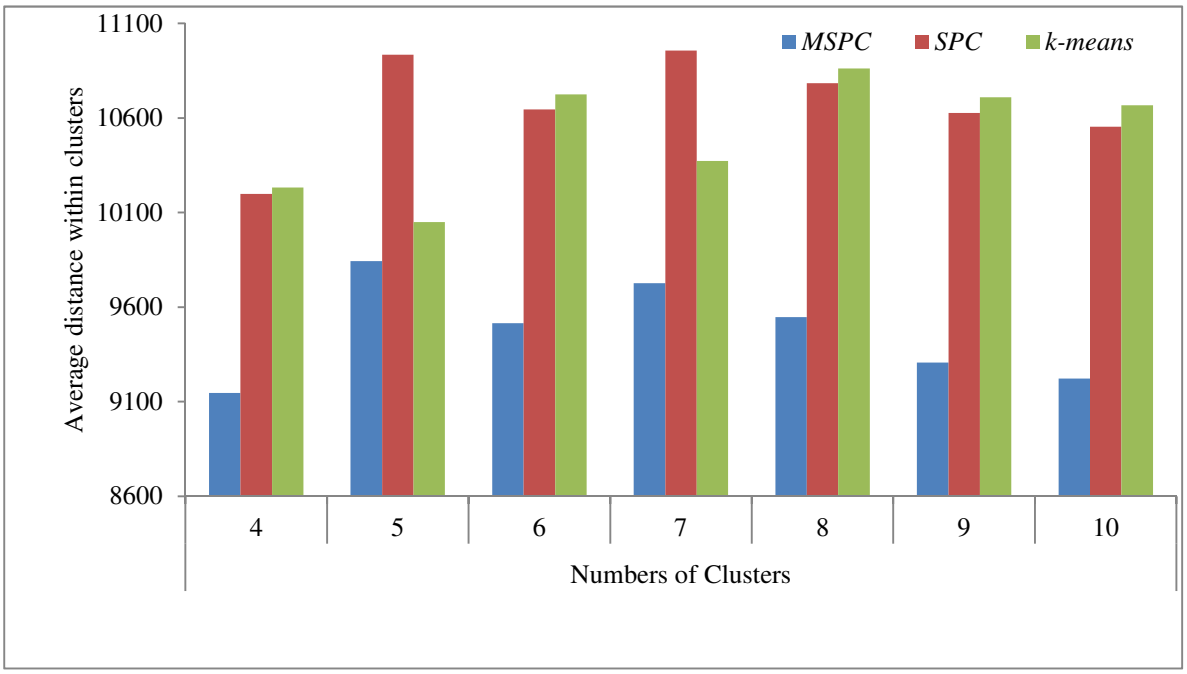


Figure 3.6: Centroid linkage compactness using mean value as a threshold

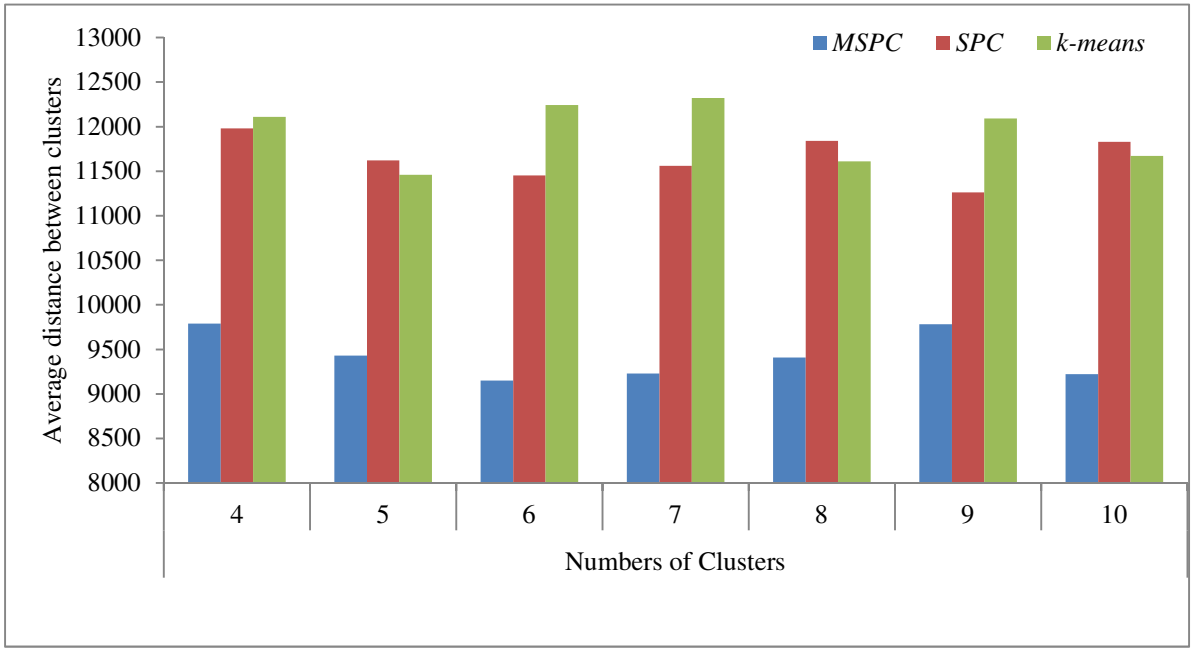


Figure 3.7: Averaged paired compactness using mean value as a threshold

It is not possible to draw conclusion about better clustering algorithm based upon compactness and separation alone. So, these algorithms are also evaluated on Dunn and *DB*

index. Dunn index is used to measure the ratio of separation to compactness. Value of this index should be large for a good clustering algorithm. Figure 3.8 shows its value for *k*-means, *SPC* and *MSPC* algorithms. From this figure, it is evident that *MSPC* algorithm provides large value of Dunn index than the *k*-means and *SPC* algorithms.

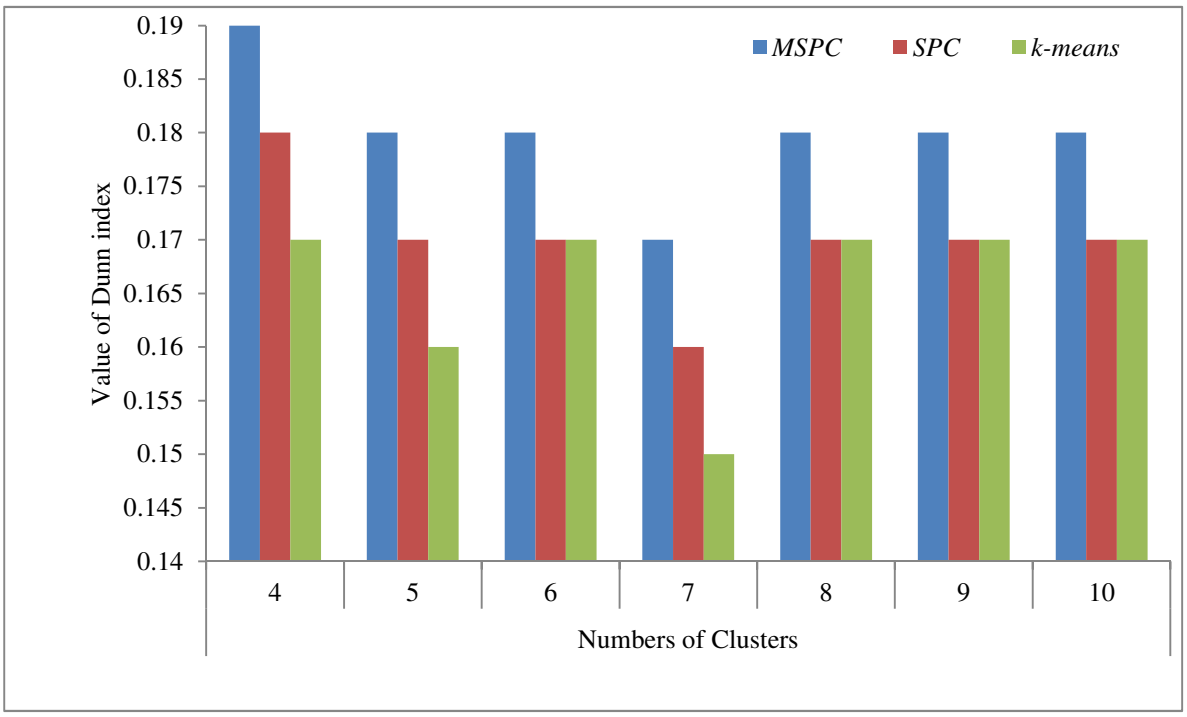


Figure 3.8: Dunn index using mean value as a threshold

DB index is used to measure the ratio of compactness to separation. Value of this index should be small for a good clustering algorithm. Figure 3.9 shows its value for *k*-means, *SPC* and *MSPC* algorithms. From this figure, it is evident that *MSPC* algorithm provides small value of *DB* index than the *k*-means and *SPC* algorithms.

For a good clustering algorithm, clusters generated by the algorithm should be as compact as possible and well separated from each other. Experiments conducted in this section on validity measures and indices confirm that *MSPC* algorithm generates well separated and compact clusters. Moreover, its performance is better than the *k*-means and *SPC* algorithms for both Dunn and *DB* validity indices. To further justify the performance of *MSPC* algorithm, threshold similarity value is taken as the median of the objects left to be clustered in the next section.

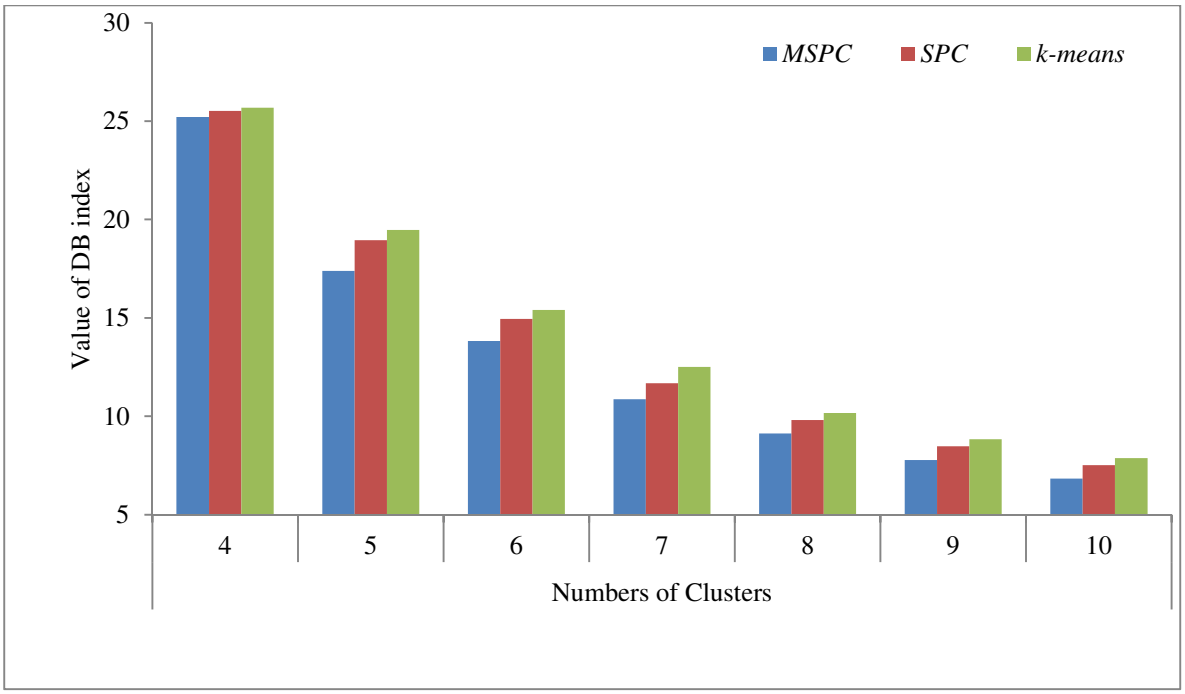


Figure 3.9: *DB* index using mean value as a threshold

3.5.1.2 Experiments using median as threshold similarity value

Figures 3.10 – 3.12 illustrate the single linkage, complete linkage and centroid linkage separation measures, respectively, for *k*-means, *SPC* and *MSPC* algorithms.

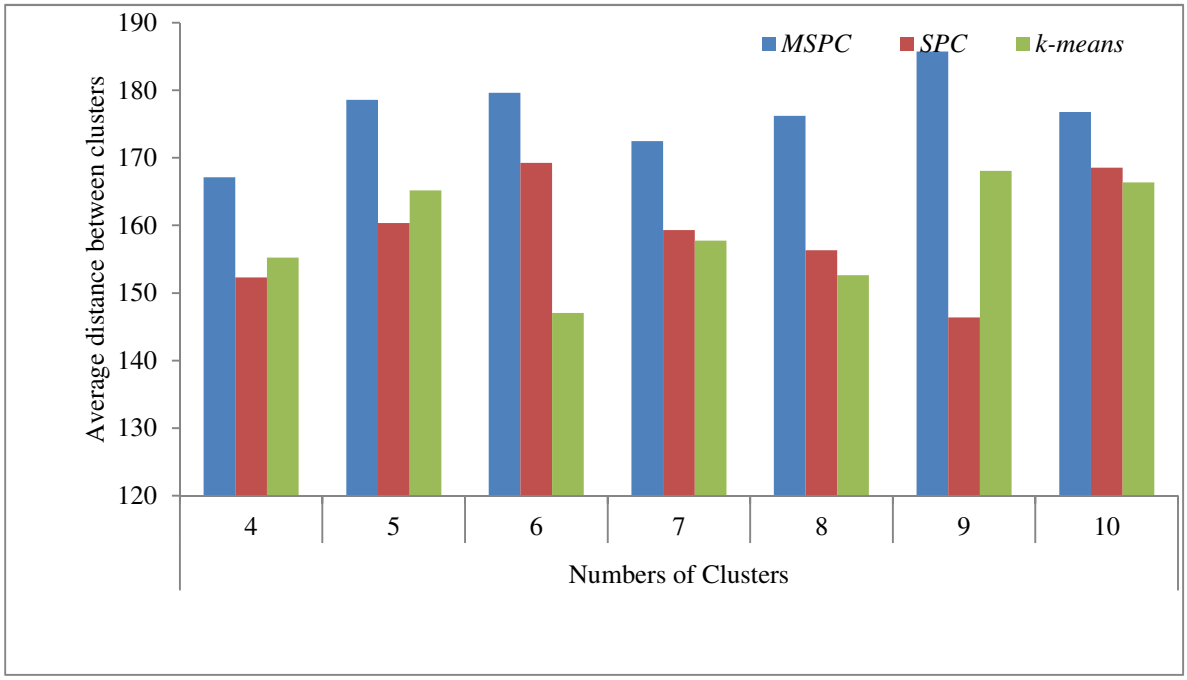


Figure 3.10: Single linkage separation using median value as a threshold

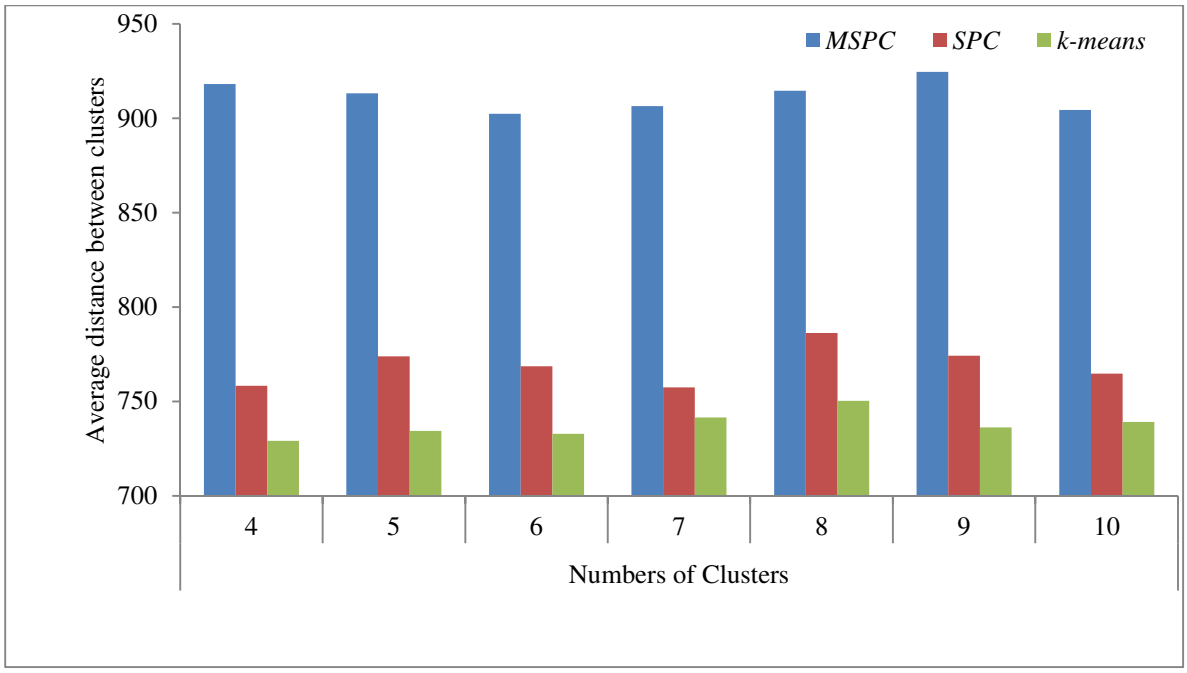


Figure 3.11: Complete linkage separation using median value as a threshold

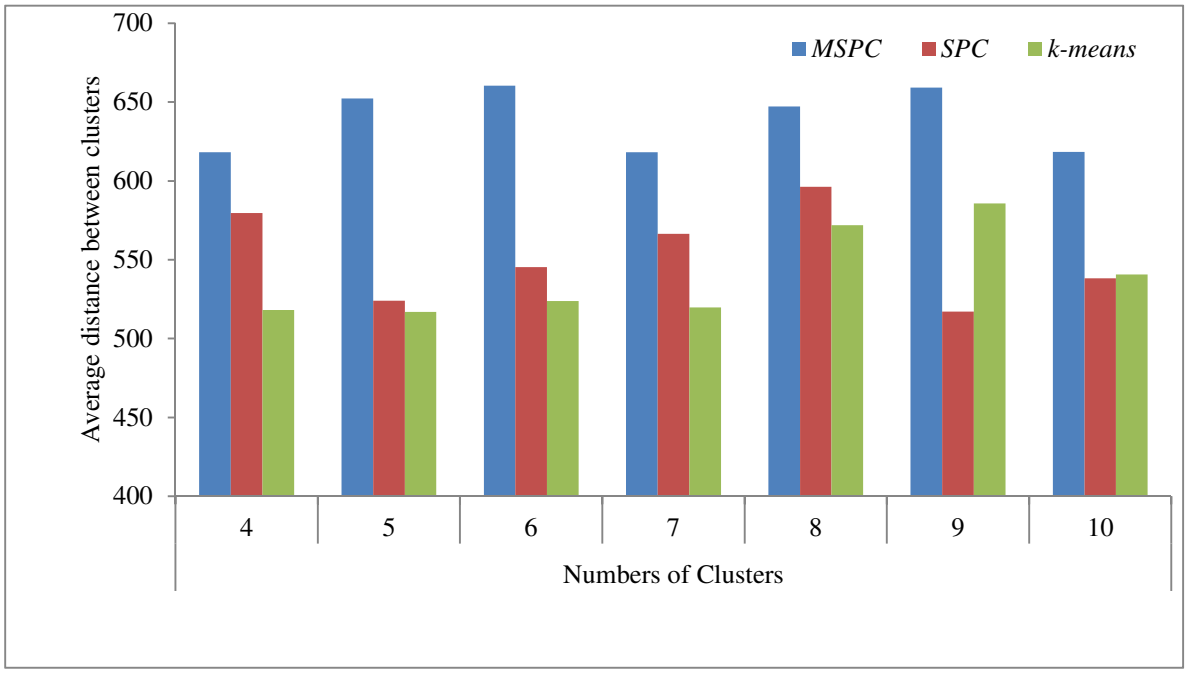


Figure 3.12: Centroid linkage separation using median value as a threshold

Figures 3.13 and 3.14 show the centroid linkage and averaged paired compactness measures, respectively, for *k*-means, *SPC* and *MSPC* algorithms. From these figures, it can also be observed that clusters generated by *MSPC* algorithm are more separate and compact than the clusters generated by *k*-means and *SPC* algorithms.

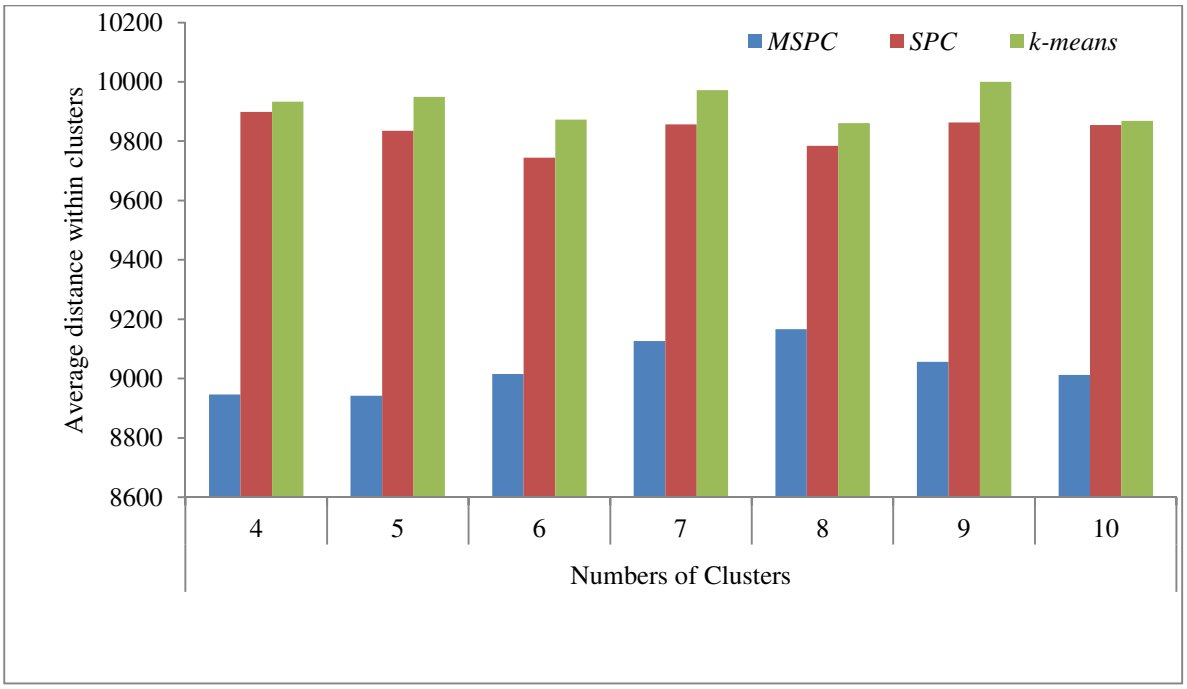


Figure 3.13: Centroid linkage compactness using median value as a threshold

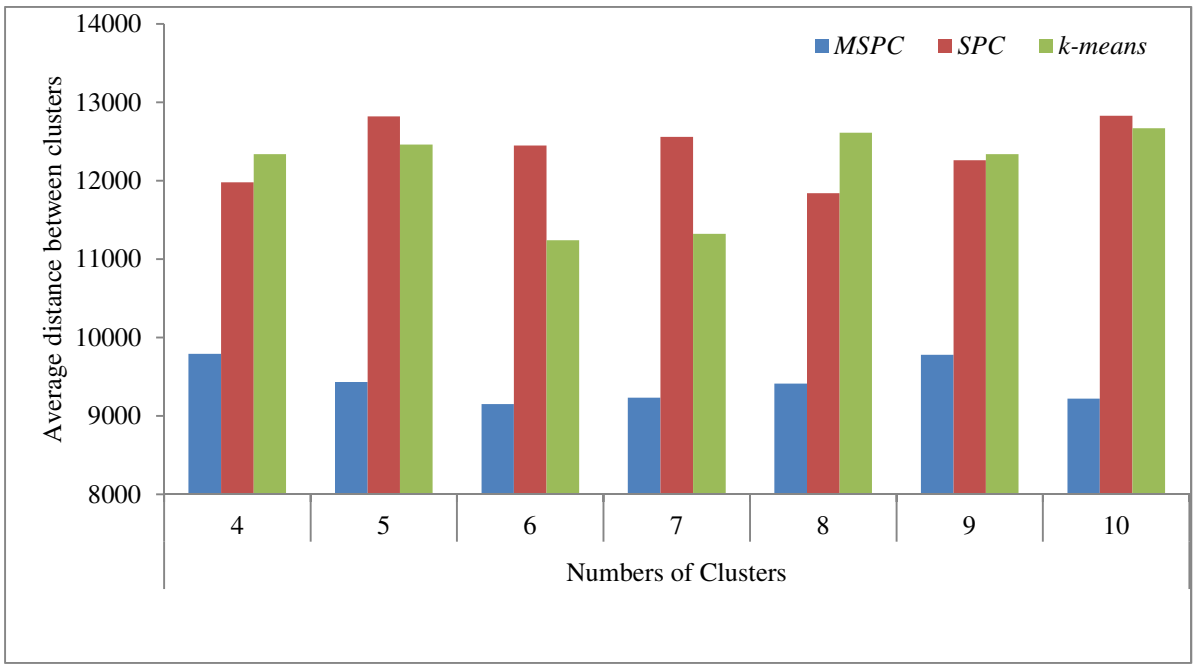


Figure 3.14: Averaged paired compactness using median value as a threshold

It is not possible to draw conclusion about better clustering algorithm based upon compactness and separation alone. So, these algorithms are also evaluated on Dunn and *DB*

index. Dunn index is used to measure the ratio of separation to compactness. Value of this index should be large for a good clustering algorithm. Figure 3.15 shows its value for *k*-means, *SPC* and *MSPC* algorithms. From this figure, it is evident that *MSPC* algorithm provides large value of Dunn index than the *k*-means and *SPC* algorithms.

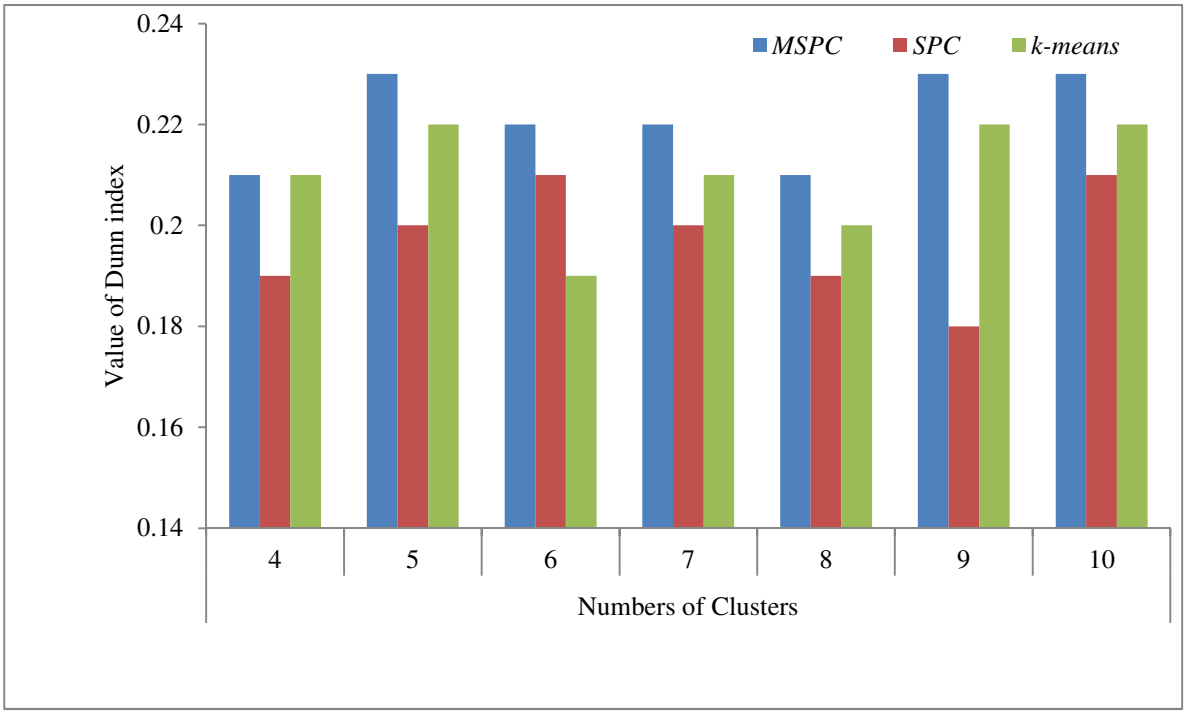


Figure 3.15: Dunn index using median value as a threshold

DB index is used to measure the ratio of compactness to separation. Value of this index should be small for a good clustering algorithm. Figure 3.16 shows its value for *k*-means, *SPC* and *MSPC* algorithms. From this figure, it is evident that *MSPC* algorithm provides small value of *DB* index than the *k*-means and *SPC* algorithms.

The experimental results obtained on artificial datasets confirm that *MSPC* algorithm is an efficient algorithm as it generates well separated and compact clusters. It also provides large Dunn index and small *DB* index values. Even though, these experiments are not the sole criteria to confirm its performance better than the *k*-means and *SPC* algorithms, it is evaluated on real datasets with threshold similarity value again taken as the mean or median of the data objects left to be clustered in the next section. In the Section 3.5.1, threshold similarity values is taken as mean value and in the Section 3.5.2, it is taken as median value.

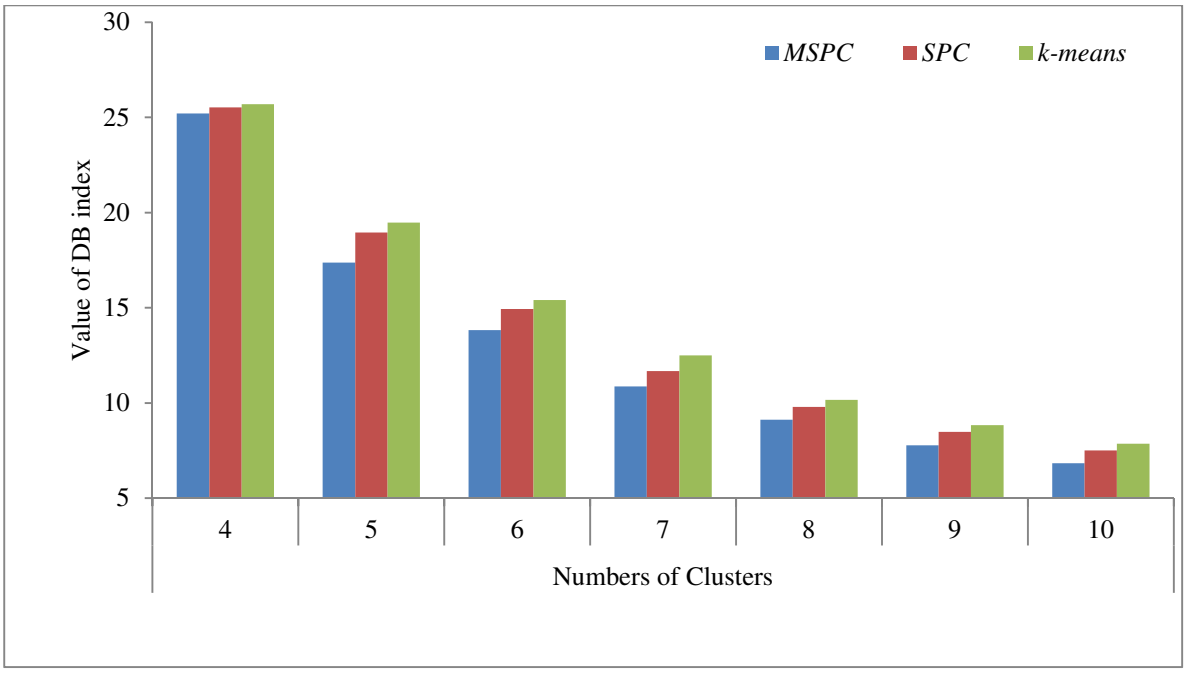


Figure 3.16: *DB* index using median value as a threshold

3.5.2 Real datasets experiments

Two experiments have been carried out on real datasets. In the first experiment, threshold similarity value is taken as the mean of paired distance of data objects and in the second experiment, it is taken as the median. In both the experiments, four real datasets: Ecoli, Iris, Seeds and Wine have been considered. These datasets are taken from the UCI repository (http://archive.ics.uci.edu/ml/datasets.html). Table 3.1 shows the characteristics of these datasets.

Table 3.1: Characteristics of real datasets

Dataset	# Data objects	# Attributes	# Clusters
Ecoli	336	7	8
Iris	150	4	3
Seeds	210	7	3
Wine	178	13	3

3.5.2.1 Experiments using mean as threshold similarity value

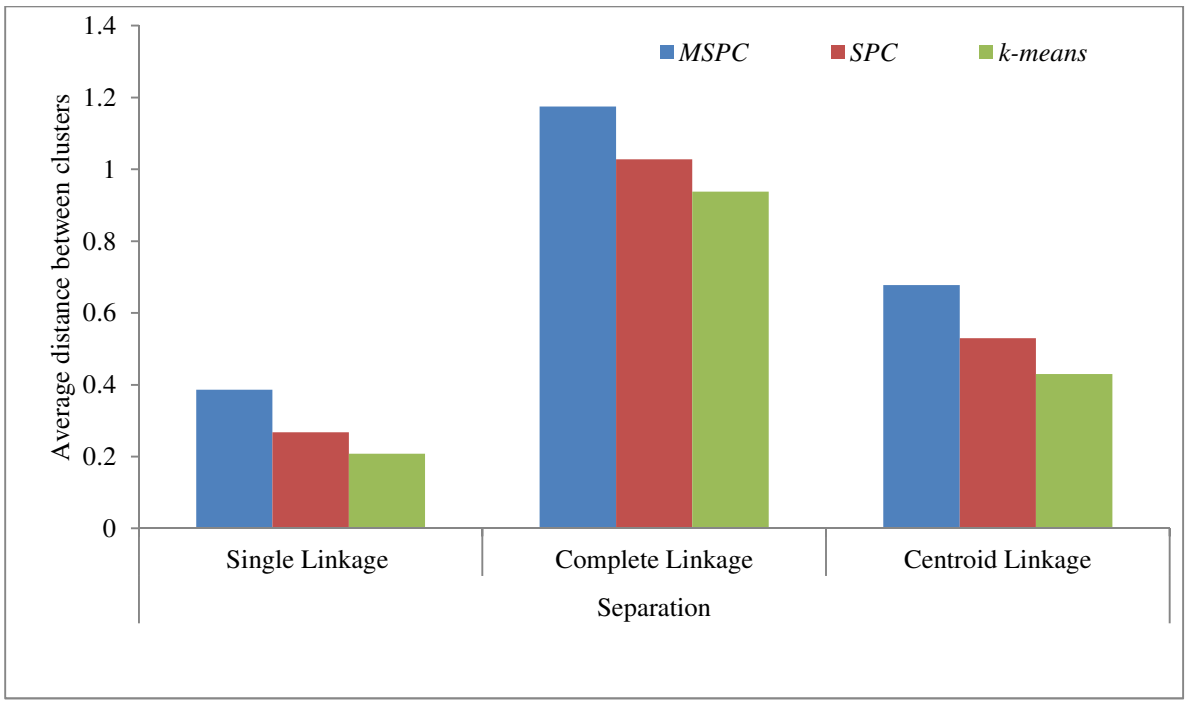
In the first experiment, threshold similarity is taken as the mean value. The *MSPC* algorithm is executed 1000 times and a varying number of clusters are generated for Ecoli, Iris, Seeds

and Wine datasets; and then the same number of clusters is generated by the *k*-means and *SPC* algorithms. Table 3.2 shows the frequency of number of clusters generated by *MSPC* algorithm.

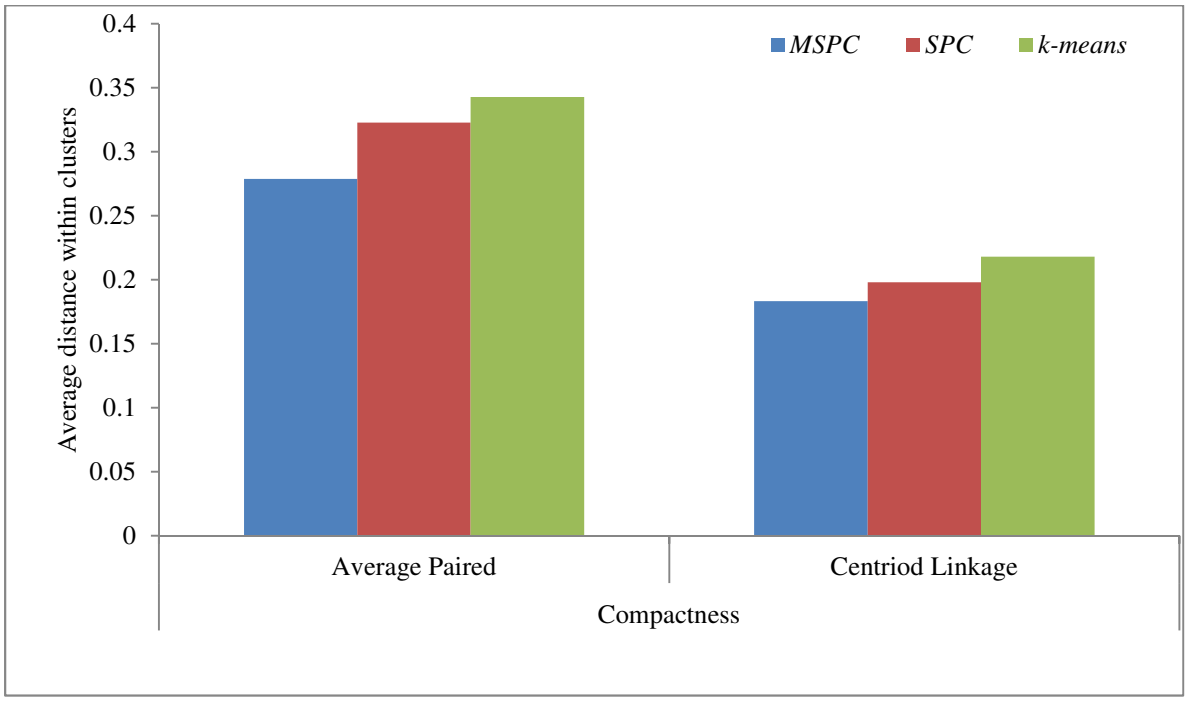
Table 3.2: Clusters generated using threshold similarity as a mean value

Dataset	# Clusters	Frequency of Generation
Ecoli	5	2
	6	111
	7	330
	8	**388**
	9	136
	10	31
	11	2
Iris	2	396
	3	**603**
	4	1
Seeds	2	35
	3	**810**
	4	154
	5	1
Wine	**3**	**691**
	4	306
	5	3

It is observed that proposed algorithm generates mostly actual number of clusters present in the dataset as shown in the table by bold faces values. These values can be verified from the characteristics of datasets shown in the Table 3.1. A comparison of all validity measures for *k*-means, *SPC* and *MSPC* algorithms on Ecoli dataset is presented in Figure 3.17 and Table 3.3 presents the relative improvement of *MSPC* algorithm for all validity measures on this dataset with respect to *k*-means and *SPC* algorithms.



(a)



(b)

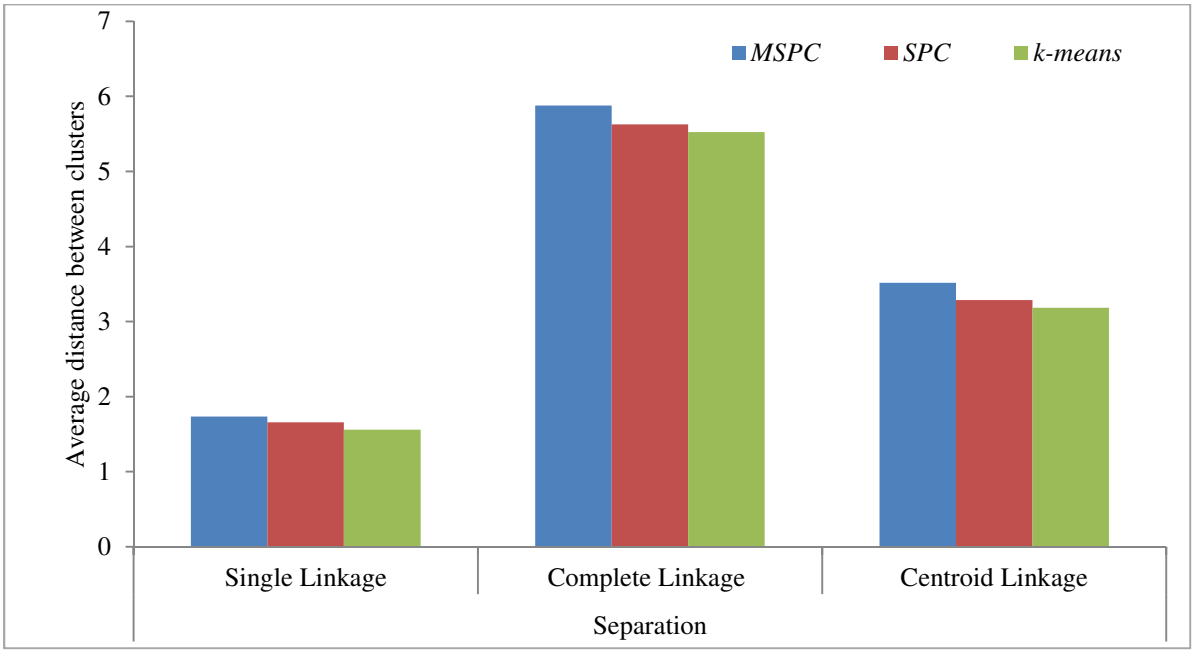
Figure 3.17: Ecoli dataset using mean as a threshold value

(a) Separation based comparison (b) Compactness based comparison

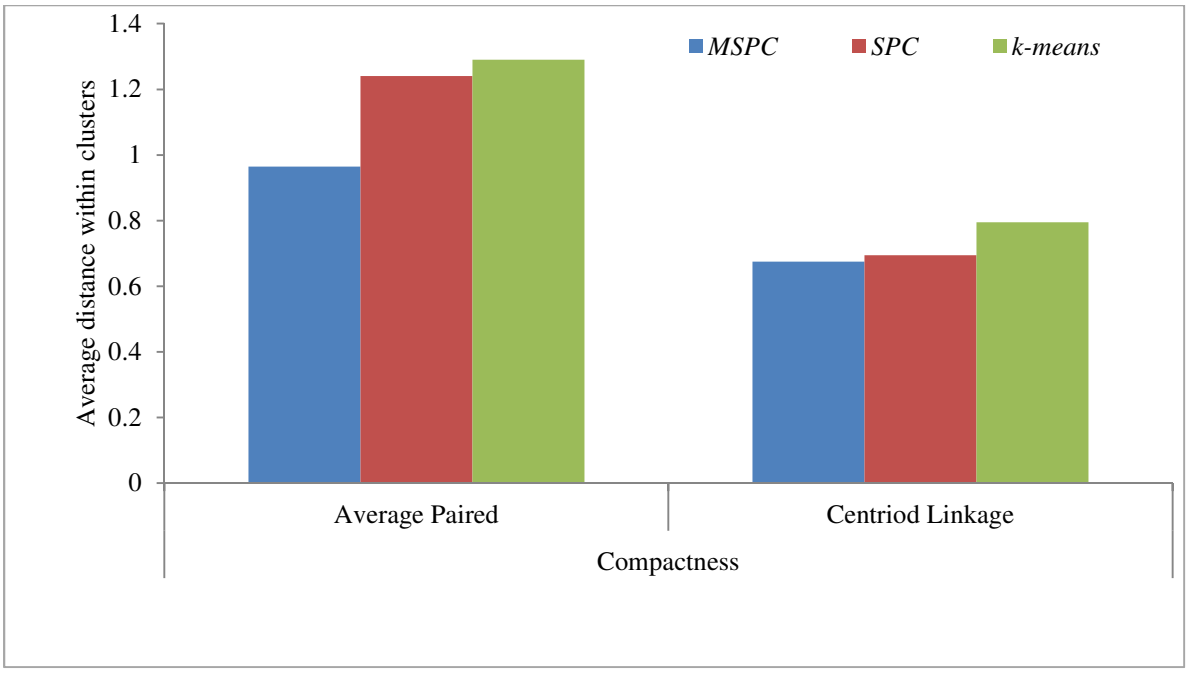
Table 3.3: Relative improvement of *MSPC* algorithm for validity measures on Ecoli dataset using mean as a threshold value

Validity Measures		*SPC*	*k*-means
Separation Methods	Single Linkage	30.5%	46.1%
	Complete Linkage	12.4%	20.1%
	Centroid Linkage	21.8%	36.5%
Compaction Methods	Averaged Paired Distance	15.7%	22.9%
	Centroid Based	8.1%	19.1%

A comparison of all validity measures for *k*-means, *SPC* and *MSPC* algorithms on Iris dataset is presented in Figure 3.18 and Table 3.4 presents the relative improvement of *MSPC* algorithm for all validity measures on this dataset with respect to *k*-means and *SPC* algorithms.



(a)



(b)

Figure 3.18: Iris dataset using mean as a threshold value

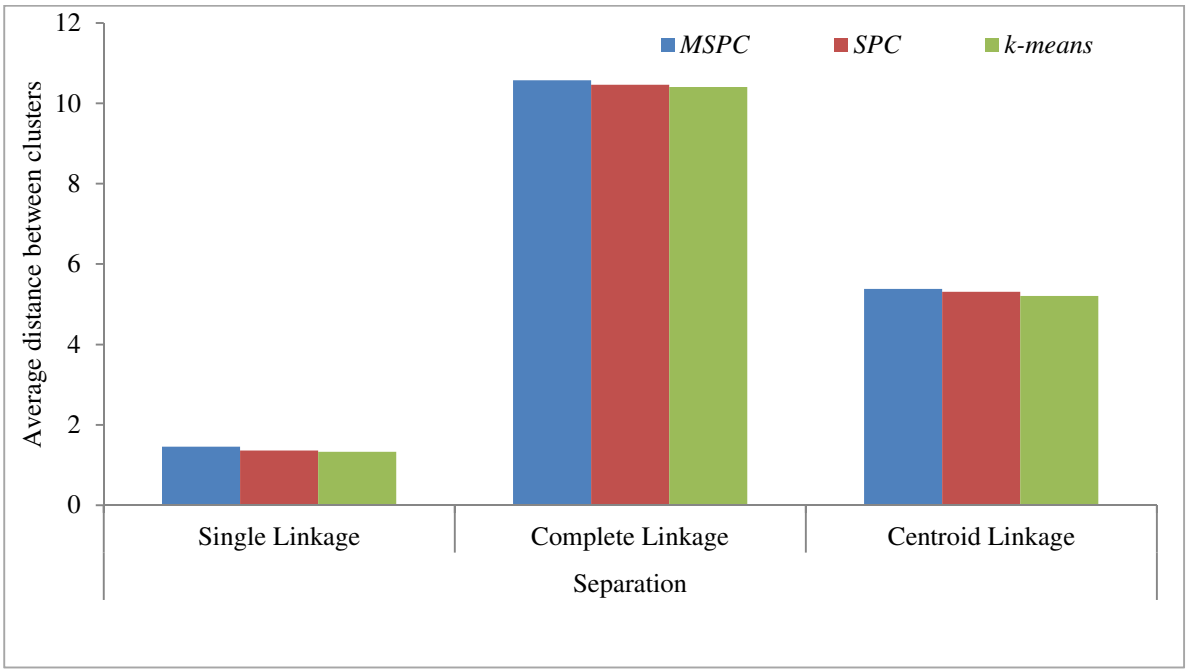
(a) Separation based comparison (b) Compactness based comparison

Table 3.4: Relative improvement of *MSPC* algorithm for validity measures on Iris dataset using mean as a threshold value

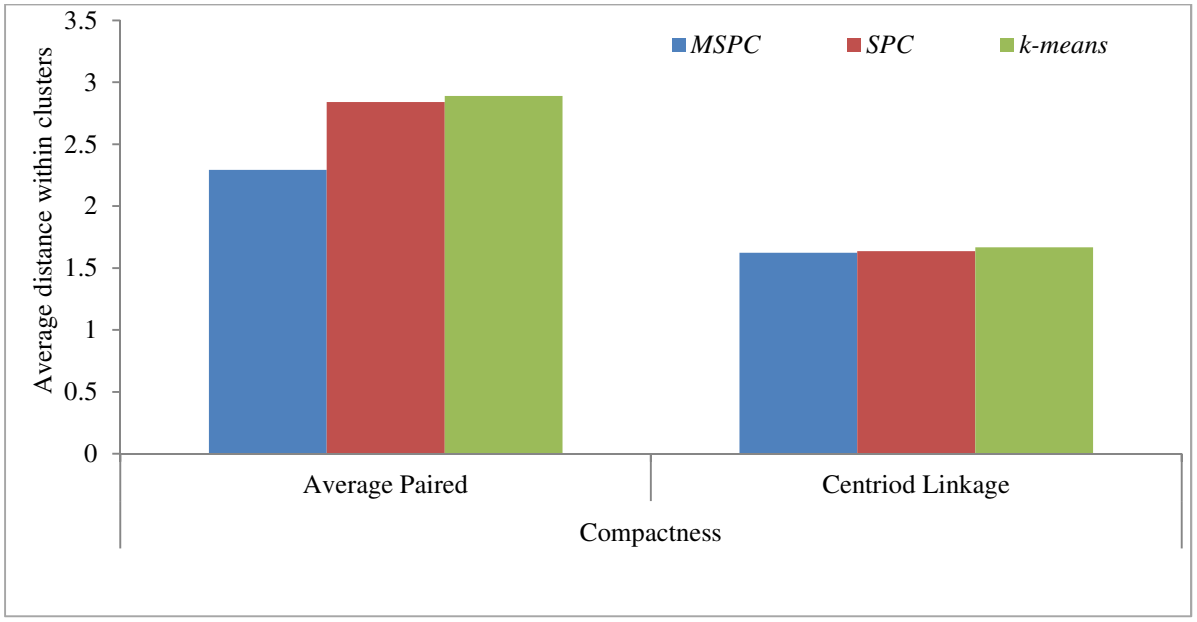
Validity Measures		*SPC*	*k*-means
Separation Methods	Single Linkage	4.4%	10.2%
	Complete Linkage	4.2%	5.9%
	Centroid Linkage	6.52%	9.3%
Compaction Methods	Averaged Paired Distance	28.5%	2.9%
	Centroid Based	33.7%	17.7%

A comparison of all validity measures for *k*-means, *SPC* and *MSPC* algorithms on Seeds dataset is presented in Figure 3.19 and Table 3.5 presents the relative improvement of *MSPC*

algorithm for all validity measures on this dataset with respect to *k*-means and *SPC* algorithms.



(a)



(b)

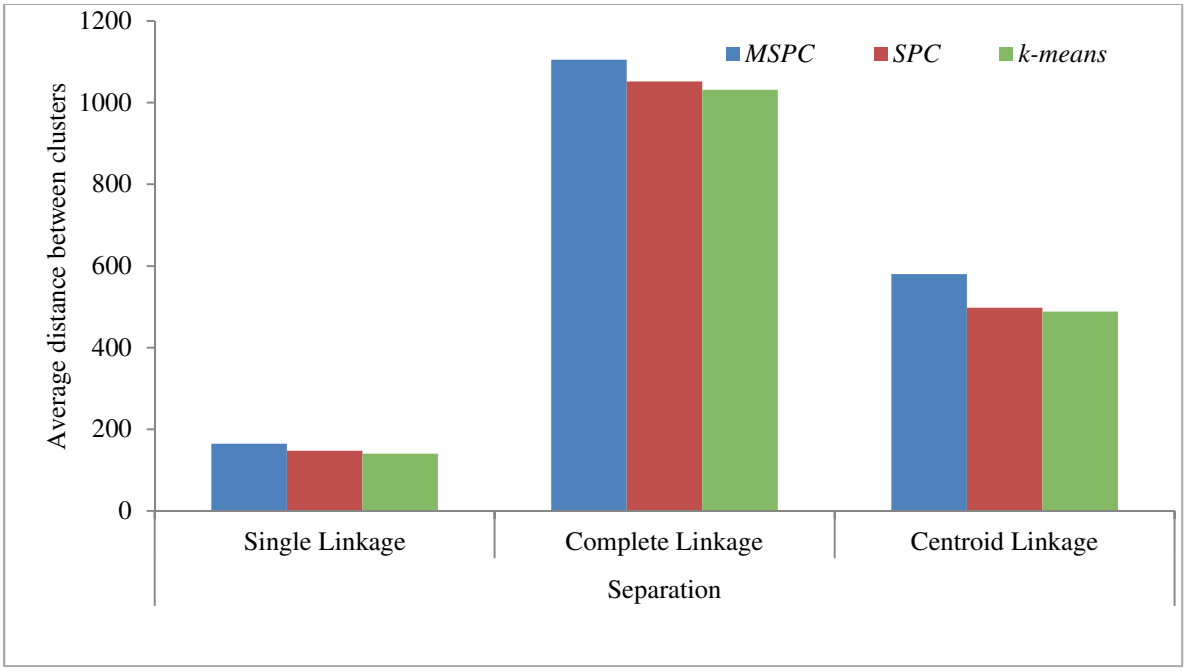
Figure 3.19: Seeds dataset using mean as a threshold value

(a) Separation based comparison (b) Compactness based comparison

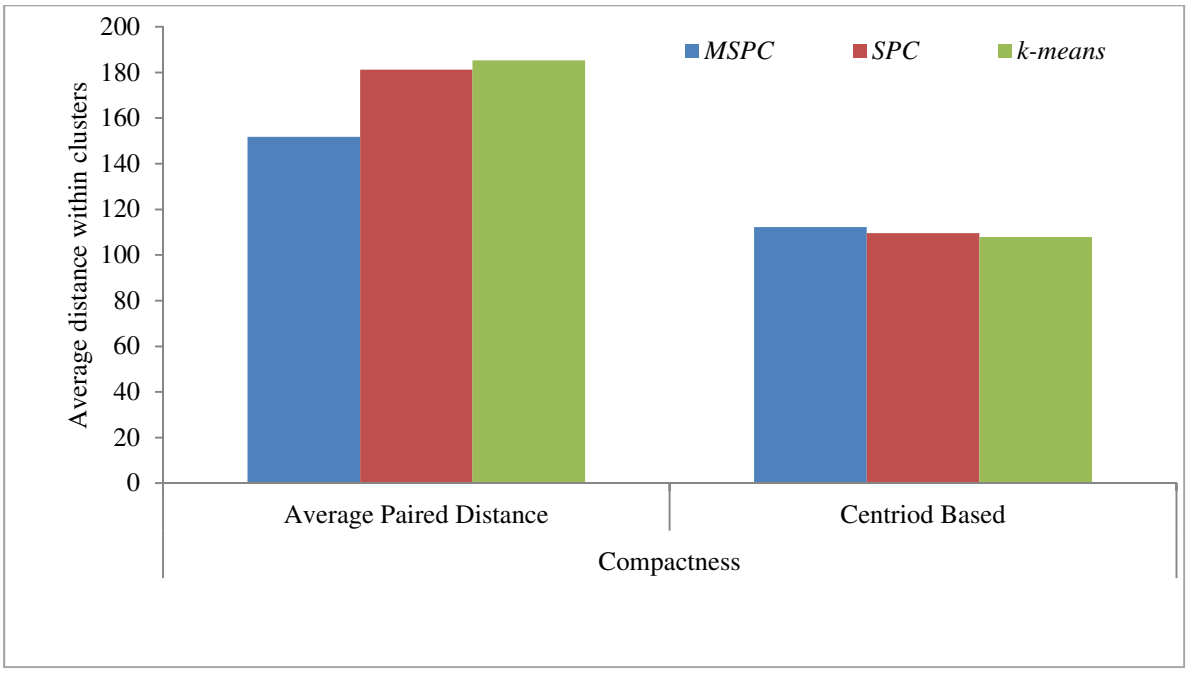
Table 3.5: Relative improvement of *MSPC* algorithm for validity measures on Seeds dataset using mean as a threshold value

Validity Measures		*SPC*	*k*-means
Separation Methods	Single Linkage	6.5%	8.6%
	Complete Linkage	1.1%	1.6%
	Centroid Linkage	1.3%	3.2%
Compaction Methods	Averaged Paired Distance	23.8%	26.1%
	Centroid Based	0.7%	2.6%

A comparison of all validity measures for *k*-means, *SPC* and *MSPC* algorithms on Wine dataset is presented in Figures 3.20 and Table 3.6 presents the relative improvement of *MSPC* algorithm for all validity measures on this dataset with respect to *k*-means and *SPC* algorithms. From Figures 3.17 - 3.20 and Tables 3.3 - 3.6, it can be observed that these experiments confirm a good performance of the *MSPC* algorithm in terms of well separated and compact clusters. Further, these algorithms are also evaluated on two validity indices: Dunn and *DB* index.



(a)



(b)

Figure 3.20: Wine dataset using mean as a threshold value

(a) Separation based comparison (b) Compactness based comparison

Table 3.6: Relative improvement of *MSPC* algorithm for validity measures on Wine dataset using mean as a threshold value

Validity Measures		*SPC*	*k*-means
Separation Methods	Single Linkage	10.1%	14.3%
	Complete Linkage	4.8%	6.6%
	Centroid Linkage	14.1%	15.8%
Compaction Methods	Averaged Paired Distance	19.4%	22.1%
	Centroid Based	2.3%	3.7%

Figure 3.21 depicts the graphical comparison of values of Dunn index and Figure 3.22 depicts the graphical comparison of values of *DB* index. From these figures, it is evident that *MSPC* algorithm provides large value of Dunn index and small value of *DB* index.

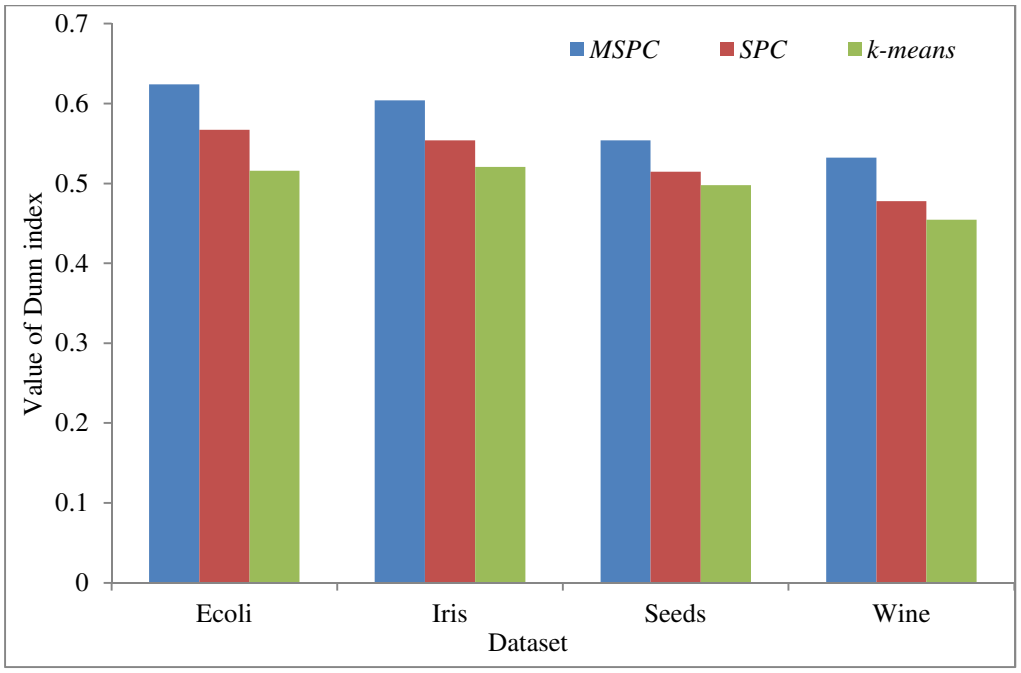


Figure 3.21: Dunn index on real datasets using mean as a threshold value

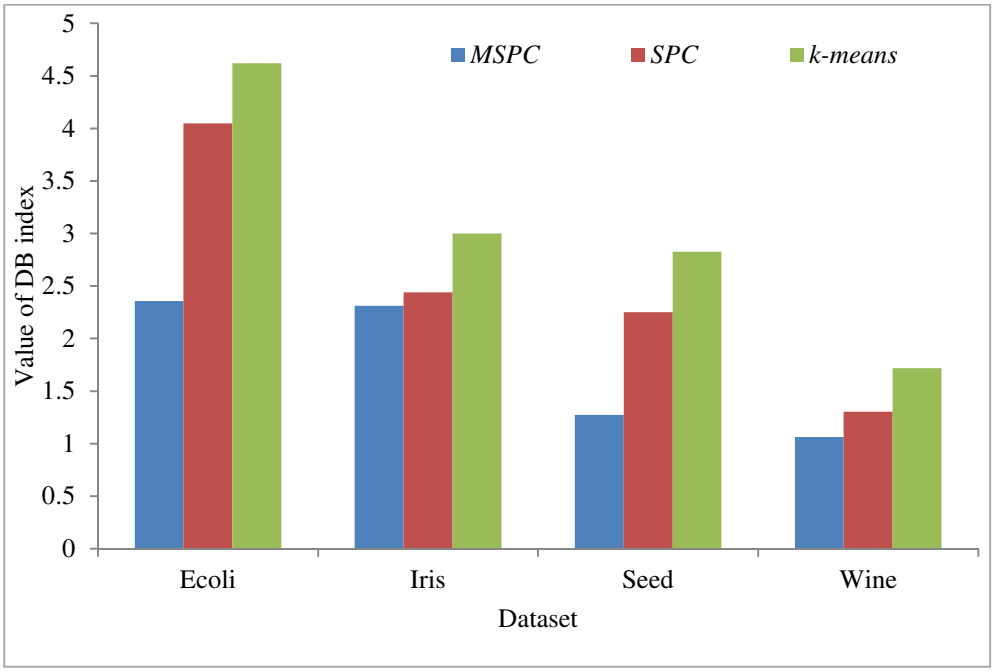


Figure 3.22: *DB* index on real datasets using mean as a threshold value

3.5.2.2 Experiments using median as threshold similarity value

In the second experiment, threshold similarity is taken as the median value. The *MSPC* algorithm is again executed 1000 times and a varying number of clusters is generated; and

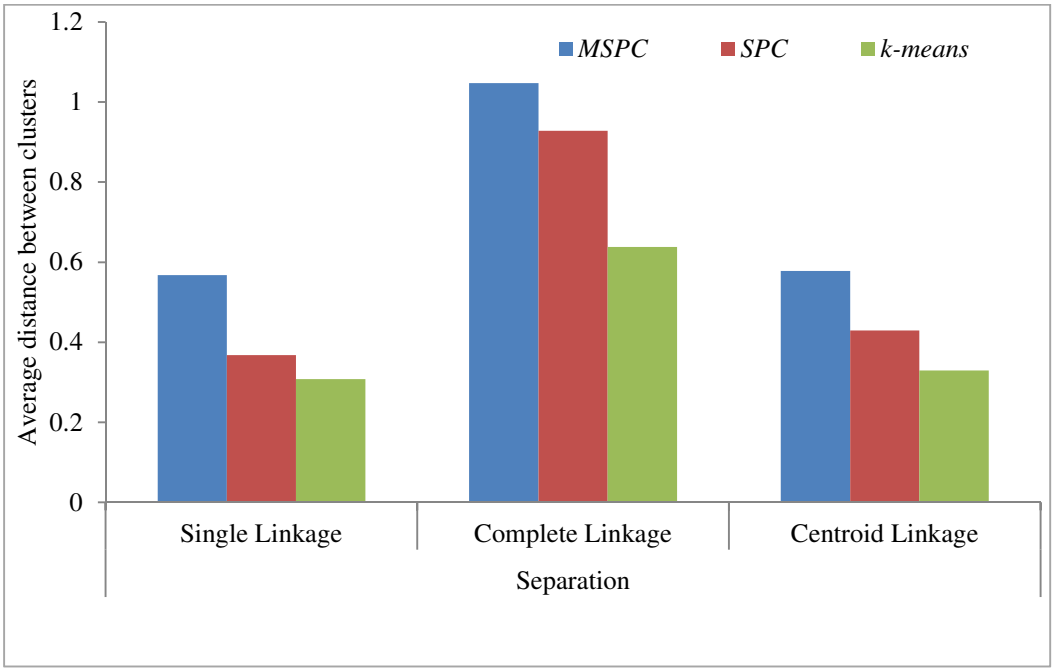
then the same number of clusters is generated by the *k*-means and *SPC* algorithms. Table 3.7 shows the frequency of number of clusters generated by *MSPC* algorithm.

Table 3.7: Clusters generated using threshold similarity as a median value

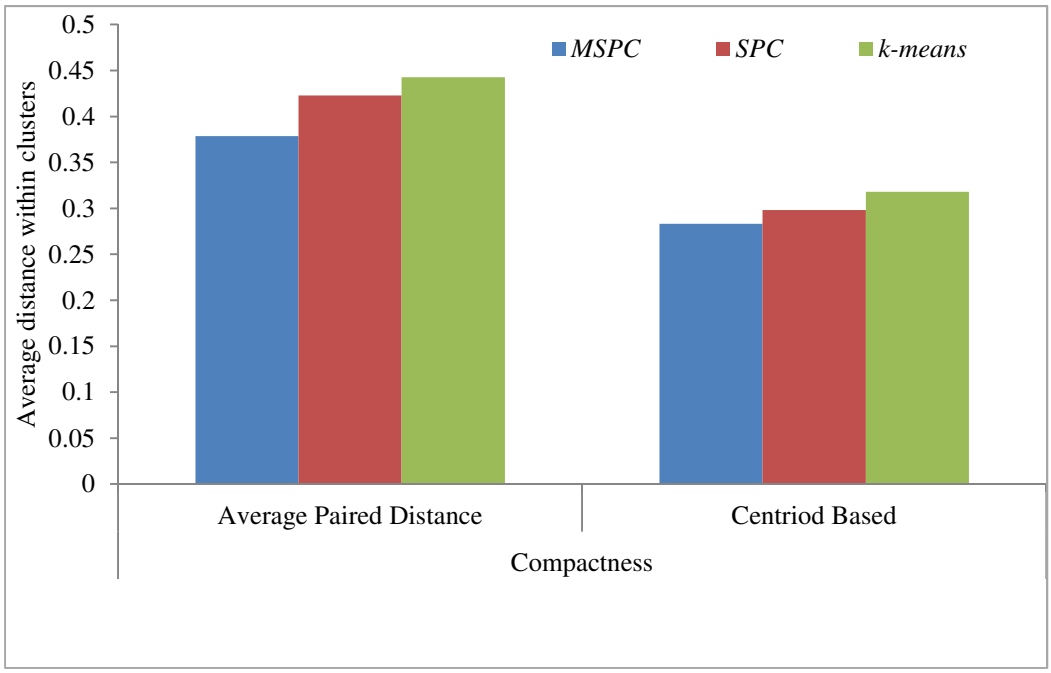
Dataset	# Clusters	Frequency of Generation
Ecoli	5	3
	6	101
	7	278
	8	**429**
	9	134
	10	50
	11	5
Iris	2	306
	3	**674**
	4	20
Seeds	2	55
	3	**729**
	4	204
	5	12
Wine	**3**	**651**
	4	342
	5	7

It can again be observed that proposed algorithm generates mostly actual number of clusters present in the dataset as shown in the table by bold faces values. These values can be verified from the characteristics of datasets shown in the Table 3.1. A comparison of all validity measures for *k*-means, *SPC* and *MSPC* algorithms on Ecoli dataset is presented in Figures

3.23 and Table 3.8 presents the relative improvement of *MSPC* algorithm for all validity measures on this dataset with respect to *k*-means and *SPC* algorithms.



(a)



(b)

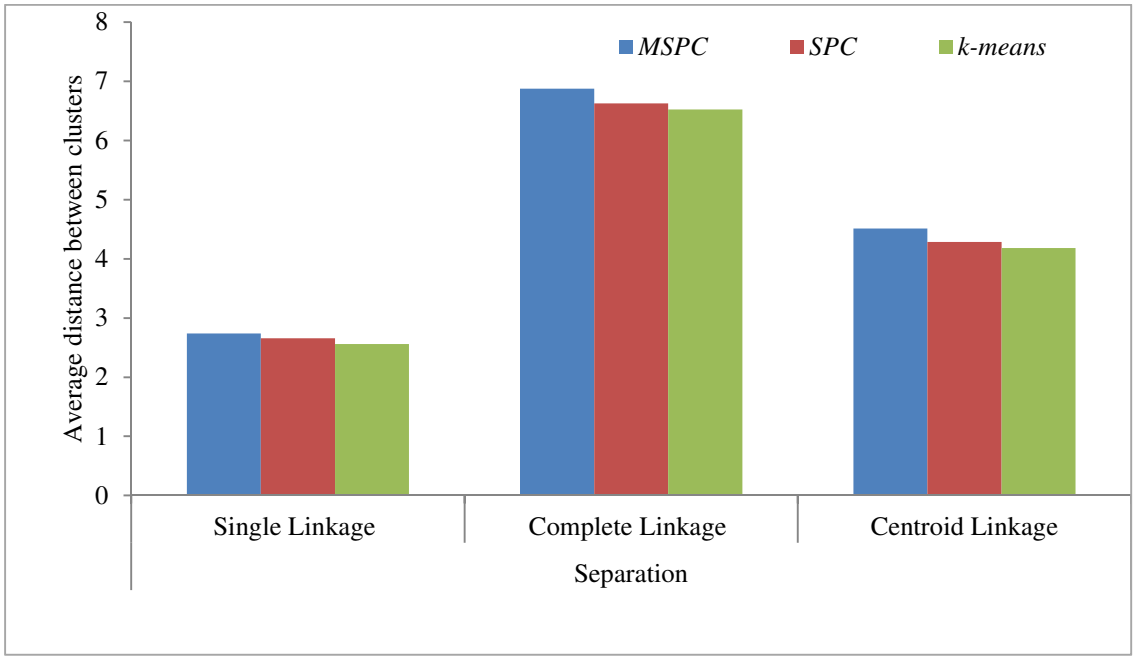
Figure 3.23: Ecoli dataset using median as a threshold value

(a) Separation based comparison (b) Compactness based comparison

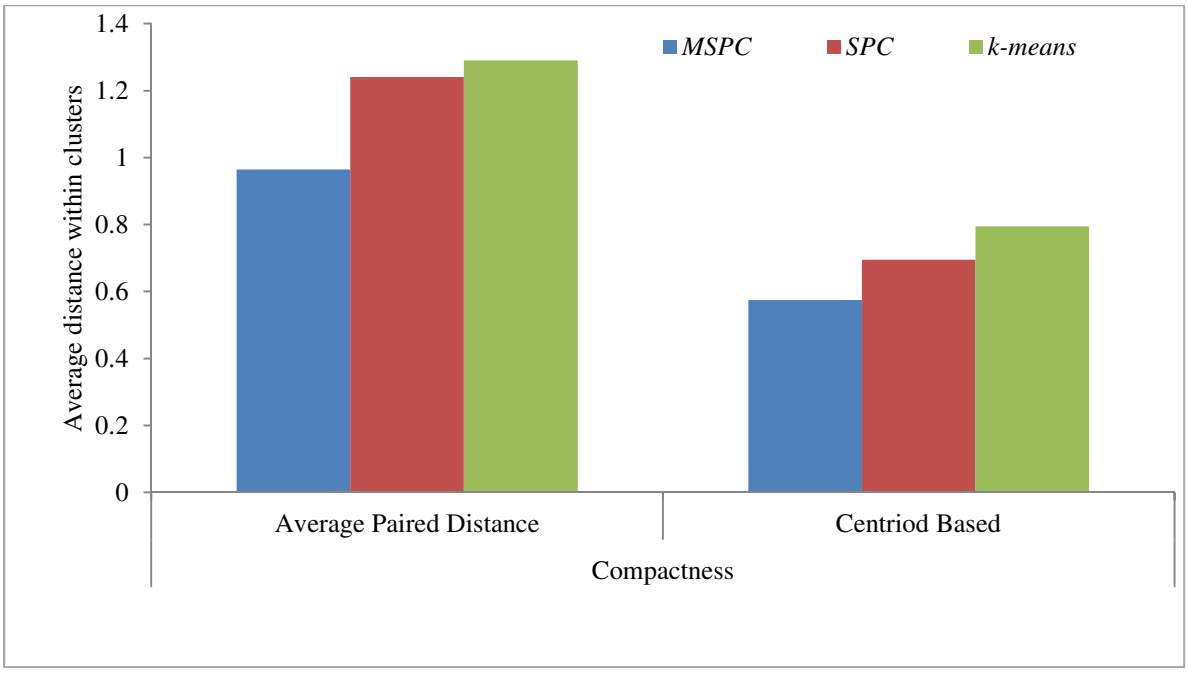
Table 3.8: Relative improvement of *MSPC* algorithm for validity measures on Ecoli dataset using median as a threshold value

Validity Measures		*SPC*	*k*-means
Separation Methods	Single Linkage	35.2%	45.7%
	Complete Linkage	11.4%	39.1%
	Centroid Linkage	25.6%	42.9%
Compaction Methods	Averaged Paired Distance	11.6%	16.8%
	Centriod Based	5.2%	12.3%

A comparison of all validity measures for *k*-means, *SPC* and *MSPC* algorithms on Iris dataset is presented in Figure 3.24 and Table 3.9 presents the relative improvement of *MSPC* algorithm for all validity measures on this dataset with respect to *k*-means and *SPC* algorithms.



(a)



(b)

Figure 3.24: Iris dataset using median as a threshold value

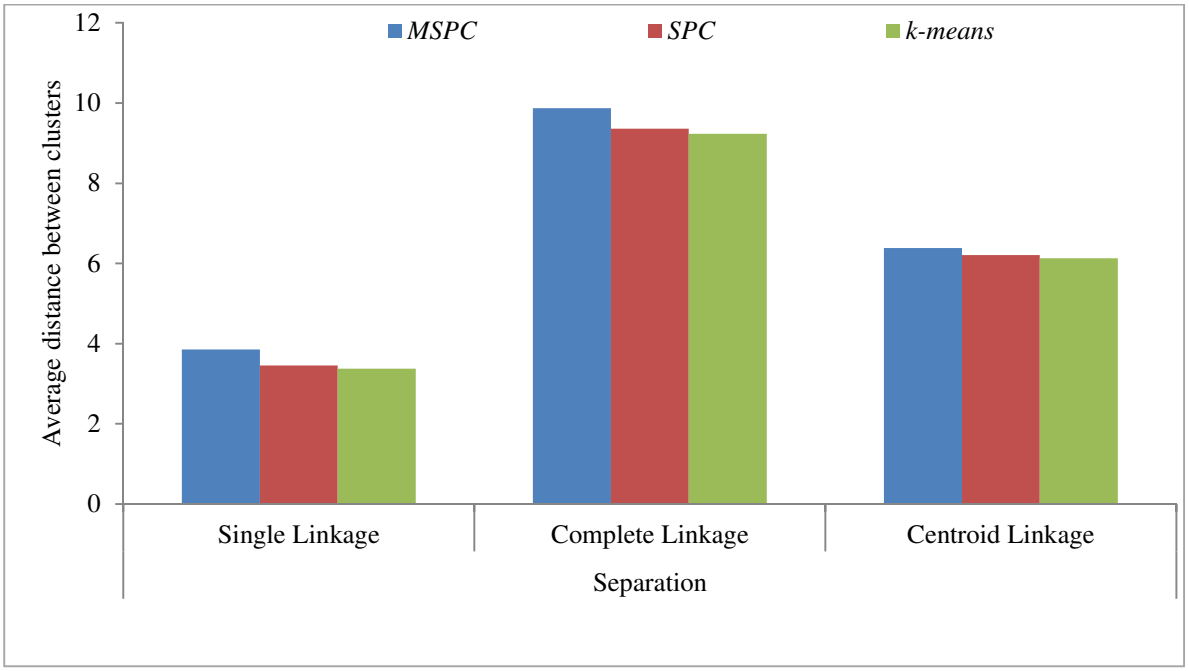
(a) Separation based comparison (b) Compactness based comparison

Table 3.9: Relative improvement of *MSPC* algorithm for validity measures on Iris dataset using median as a threshold value

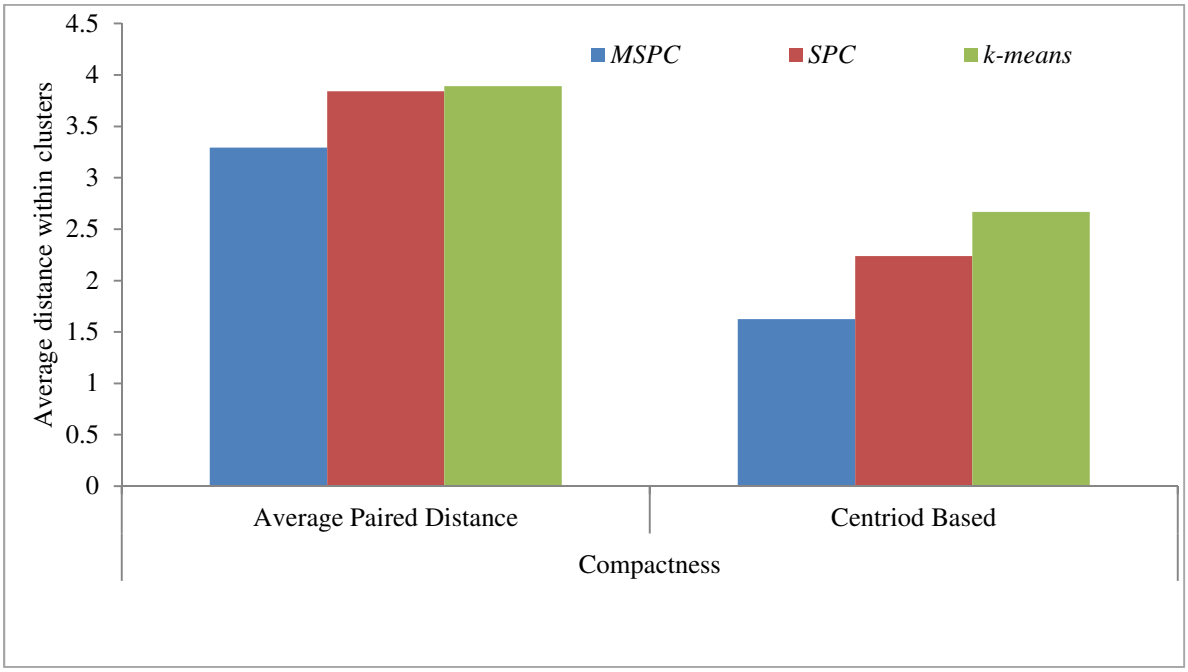
Validity Measures		*SPC*	*k*-means
Separation Methods	Single Linkage	2.8%	6.4%
	Complete Linkage	3.6%	5.1%
	Centroid Linkage	5.0%	7.2%
Compaction Methods	Averaged Paired Distance	28.5%	33.7%
	Centriod Based	20.8%	38.2%

A comparison of all validity measures for *k*-means, *SPC* and *MSPC* algorithms on Seeds dataset is presented in Figure 3.25 and Table 3.10 presents the relative improvement of

MSPC algorithm for all validity measures on this dataset with respect to *k*-means and *SPC* algorithms.



(a)



(b)

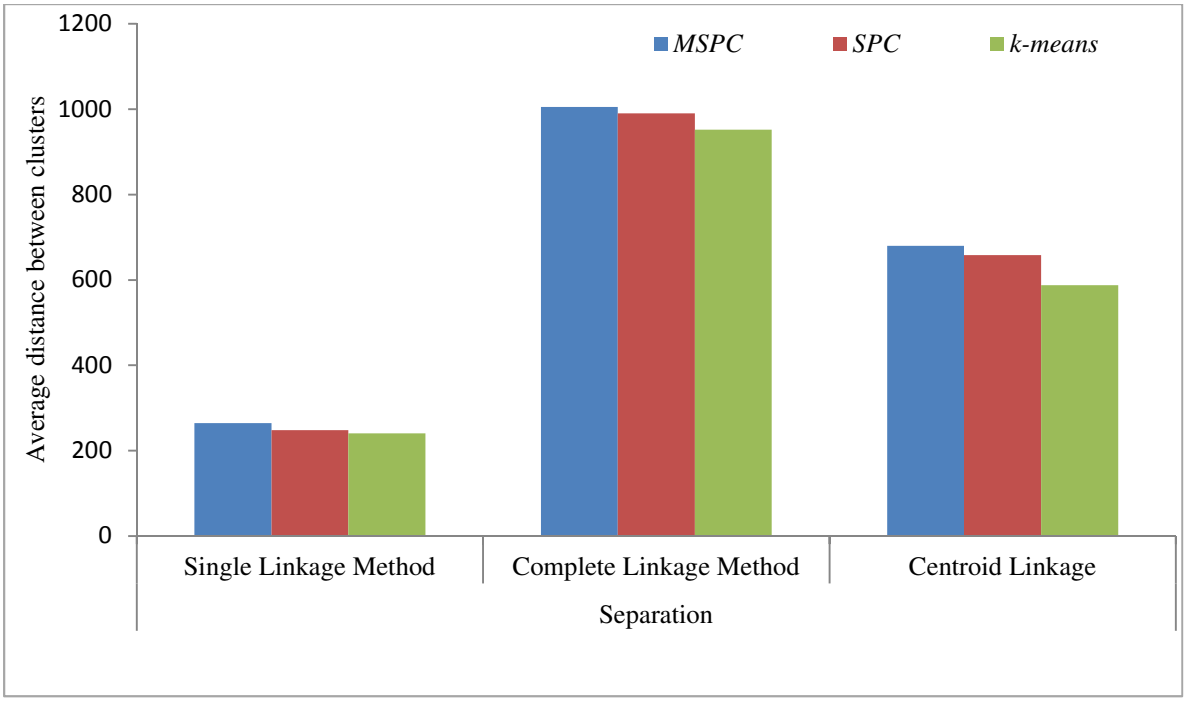
Figure 3.25: Seeds dataset using median as a threshold value

(a) Separation based comparison (b) Compactness based comparison

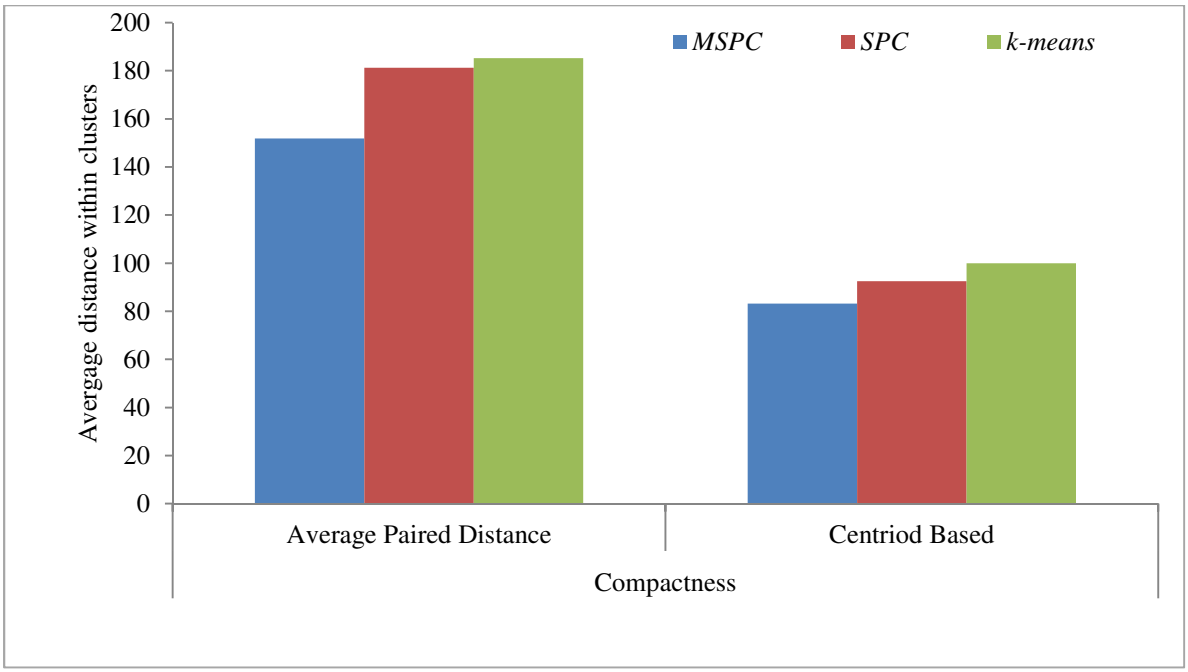
Table 3.10: Relative improvement of *MSPC* algorithm for validity measures on Seeds dataset using median as a threshold value

Validity Measures		*SPC*	*k*-means
Separation Methods	Single Linkage	10.2%	12.5%
	Complete Linkage	5.1%	6.4%
	Centroid Linkage	2.7%	4.1%
Compaction Methods	Averaged Paired Distance	16.6%	18.1%
	Centriod Based	37.7%	64.2%

A comparison of all validity measures for *k*-means, *SPC* and *MSPC* algorithms on Wine dataset is presented in Figure 3.26 and Table 3.11 presents the relative improvement of *MSPC* algorithm for all validity measures on this dataset with respect to *k*-means and *SPC* algorithms.



(a)



(b)

Figure 3.26: Wine dataset using median as a threshold value

(a) Separation based comparison (b) Compactness based comparison

Table 3.11: Relative improvement of *MSPC* algorithm for validity measures on Wine dataset using median as a threshold value

Validity Measures		*SPC*	*k*-means
Separation Methods	Single Linkage	6.3%	8.9%
	Complete Linkage	1.4%	5.2%
	Centroid Linkage	3.2%	13.5%
Compaction Methods	Averaged Paired Distance	19.4%	22.1%
	Centriod Based	11.2%	20.1%

From Figures 3.23 - 3.26 and Tables 3.8 - 3.11, it can be observed that *MSPC* algorithm performs better than *k*-means and *SPC* algorithms. Further, these algorithms are also evaluated on two validity indices: Dunn and *DB* index. Figure 3.27 depicts the graphical

comparison of values of Dunn index and Figure 3.28 depicts the graphical comparison of values of *DB* index. From these figures, it is evident that *MSPC* algorithm provides large value of Dunn index and small value of *DB* index.

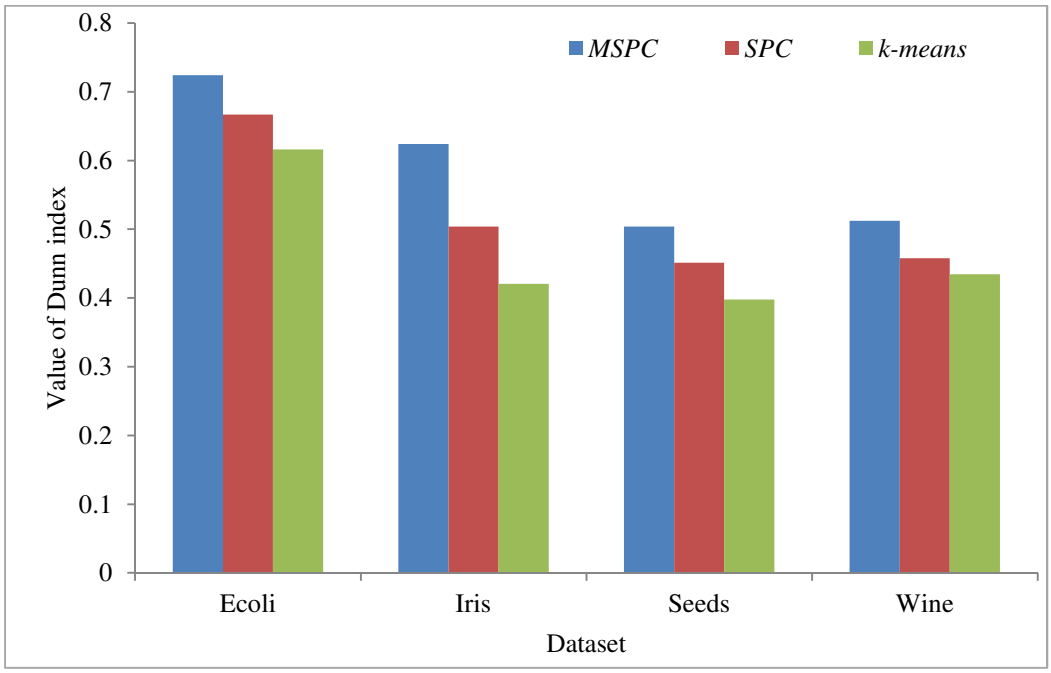


Figure 3.27: Dunn index on real datasets using median as a threshold value

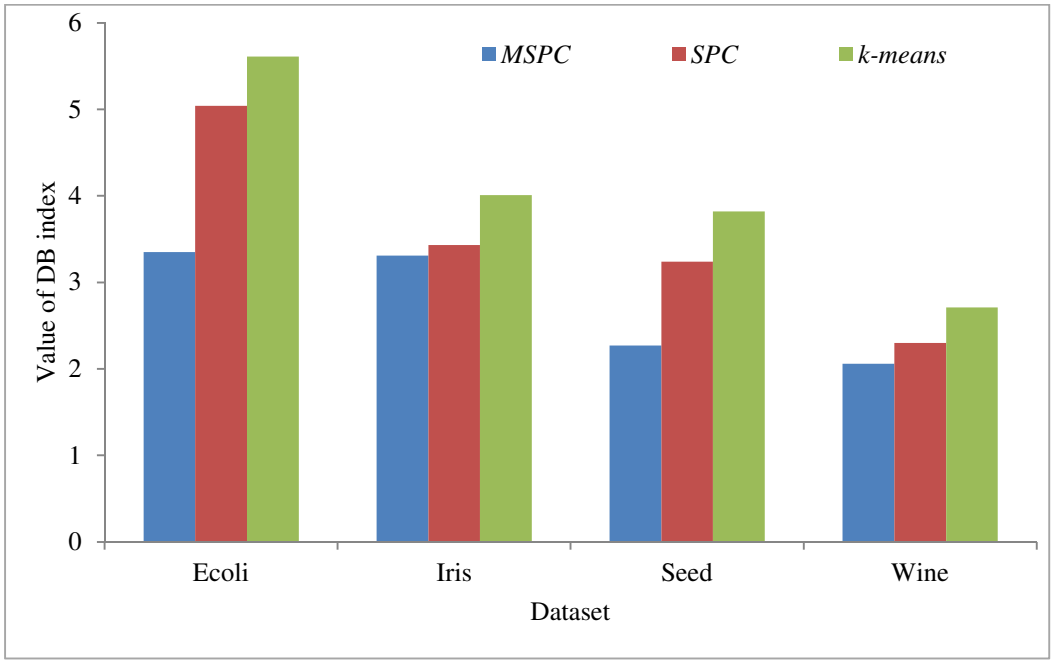


Figure 3.28: *DB* index on real datasets using median as a threshold value

MSPC algorithm has been implemented using mean or median as the threshold similarity value. In both cases, it gives better results than existing SPC and k-means algorithms. While using median as the threshold similarity value it gives 56.0% better results than using mean as threshold similarity value. This is worth mentioning here that the proposed algorithm, MSPC, does not require user defined parameters. It, however, is sensitive to the order of selection of data objects. Due to this limitation, it generates variable number of clusters on successive runs.

Chapter Summary

In this chapter, a modified single pass clustering algorithm has been proposed. It uses a threshold similarity value as mean/median of the paired distance of all data objects left to be clustered. Thus, it relinquishes the requirement of user specified parameters. Performance evaluation of clustering algorithms is one of the most important issues in cluster analysis to justify the selection of appropriate algorithm for clustering. The proposed algorithm has been compared with existing *k*-means and *SPC* algorithms on artificial and real datasets. Performance of these algorithms is validated for existing validity measures and indices. From the experiments, it has been observed that proposed algorithm generates well separated and compact clusters. Moreover, it produces mostly the actual number of clusters present in the datasets.

Chapter 4

Adaptive Threshold Based Clustering Algorithm

Partitioning based clustering methods are dependent on the user defined parameters like number of clusters and threshold similarity value. A modified single pass clustering algorithm has been discussed in the previous chapter which relinquishes these parameters. But, it has limitation of producing non-deterministic results, as it is sensitive to the order of selection of data objects. A deterministic algorithm has been proposed in this chapter for efficient clustering. It has been evaluated on artificial and real datasets. It has also been compared with *k*-means algorithm.

4.1 Introduction

Clustering is a process of finding groups from the unlabeled data objects. *k*-means algorithm, one of the clustering algorithms, is widely accepted in all application areas. But, this algorithm is non-deterministic as it requires user specified initial centroids; it is sensitive to outliers; and it generates non-overlapped clusters. However, these aspects of *k*-means have not restricted the use of this algorithm; rather, motivated researchers for improving its functionality. *k*-means, *SPC* and *MSPC* algorithms are based on the random selection of either the initial centroids or the data objects. Due to this, these algorithms produce varying results on successive runs, so these are considered non-deterministic algorithms. In this work, an efficient, *A*daptive *T*hreshold based *C*lustering (*ATC*) algorithm has been proposed which is deterministic in nature; it generates both overlapped and non-overlapped clusters; it uses an adaptive threshold similarity value; and it detects outliers.

4.2 Basic Assumption and Prerequisite for *ATC* Algorithm

The proposed algorithm works on the assumption that a given dataset can be partitioned into at least two clusters. In other words, farthest data objects of a given dataset are always the members of two different clusters; otherwise, all given data objects are in a single cluster.

There is a need of calculating distance between pair of data objects at many points during the execution of *ATC* algorithm. *So, an adjacency matrix is computed to store pair-wise distance between data objects*.

4.3 Parameters used in *ATC* Algorithm

Parameters play an important role in clustering. The functionality of clustering algorithms depends on them, so they should be specified cautiously by the users. In the *ATC* algorithm, two parameters have been introduced: neighborhood distance and minimum support. Details of these parameters are given below.

4.3.1 Neighborhood distance parameter

The first parameter used in the *ATC* algorithm is the neighborhood distance. In this algorithm, it is also known as threshold similarity value. It is used to identify all the neighbors of a data object. Existing partitioning based algorithms also employ this parameter but as a non-adaptive and user specified parameter. In the *ATC* algorithm, it is an adaptive parameter as well as not specified by the user; beside this, the value of this parameter increases during the formation of a cluster. Initially, farthest data objects, p and q, are identified from the given dataset. One of the farthest data objects is selected to initiate the formation of first cluster as explained in section 4.5. To form a cluster, *ATC* algorithm requires *N*eighborhood *D*istance (*ND*) value. Mathematically, *ND* is defined as:

$$ND = max(min_p, min_q)$$

where

$$min_p = min\{|p - r| : r \in set\ of\ un - clustered\ data\ objects\}$$

and

$$min_q = min\{|q - r| : r \in set\ of\ un - clustered\ data\ objects\}$$

Here, p and q are farthest data objects in the un-clustered data objects.

In section 4.4, it is also shown that how the value of *ND* increases during the formation of clusters.

4.3.2 Minimum support parameter

To improve the clustering results, *ATC* algorithm uses a parameter, namely, minimum support. This parameter indicates the minimum number of data objects in a cluster. The concept of using minimum support has been adopted from the association rule mining technique. In this technique, minimum support is specified by the user well before the

generation of frequent itemsets, but in the *ATC* algorithm, it can be specified by the user after generating clusters. It improves the clustering results by removing the small size clusters.

4.4 Adaptive Property of Neighborhood Distance Parameter

In the *ATC* algorithm, *ND* is not a fixed value; rather, it is increased by a small value (δ) during the formation of a cluster. The small increment to be made in *ND* is the distance between closest members of a cluster. It helps in increasing the size of a cluster. It is defined mathematically as:

$$\delta = min\{|u - v| : u, v \in cluster\}$$

Moreover, value of *ND* is initialized every time a new cluster is being formed.

4.5 Steps of *ATC* Algorithm

ATC algorithm consists of the following steps.

a) Initially, farthest data objects are identified from the given dataset.
b) Now, the distance of the closest data object to one of the farthest data objects and the distance of the closest data object to the other farthest data object are calculated. The formation of a new cluster starts from one of the farthest data objects whose closest data object is closer than its counterpart.
c) Initialize the value of *ND* and assign the selected farthest data object and all the data objects within the neighborhood distance as members of the cluster being formed.
d) Increase the value *ND* by a small value δ (as discussed in section 4.4) and assign more data objects as the members of cluster being formed. This step is repeated till new data objects are added in the cluster.
e) If more than one data object are left un-clustered then again find the farthest data objects from the un-clustered data objects and repeat the steps b), c) and d).

4.6 Illustrating the Cluster Formation Process in *ATC* Algorithm

The basic idea of cluster formation process in *ATC* algorithm is illustrated using Figures 4.1 - 4.4. Figure 4.1 shows the artificial data objects that need to be clustered. In this dataset, data objects *p* and *q* are the farthest data objects. Let min_p be the distance of closest data object to *p* and min_q be the distance of closest data object to *q* as shown in the Figure 4.2. It can be

noted from this figure that min_p is less than min_q. So, the formation of first cluster starts with the data object *p* and neighborhood distance value min_q (Figure 4.3). Data object *p* and its neighboring data objects within *ND* are assigned as members of the first cluster. Thereafter, value of *ND* is increased by a small value δ to add more data objects as members of this cluster (Figure 4.4). Value of *ND* is increased till new data objects are added in the cluster. This cluster formation process is repeated till all the data objects are clustered.

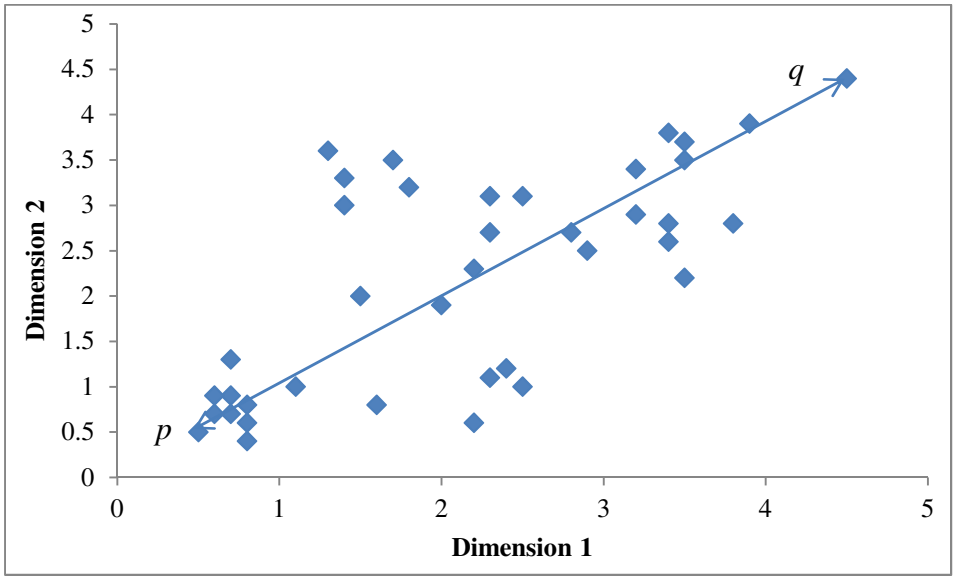


Figure 4.1: Data objects and two farthest objects *p* and *q*

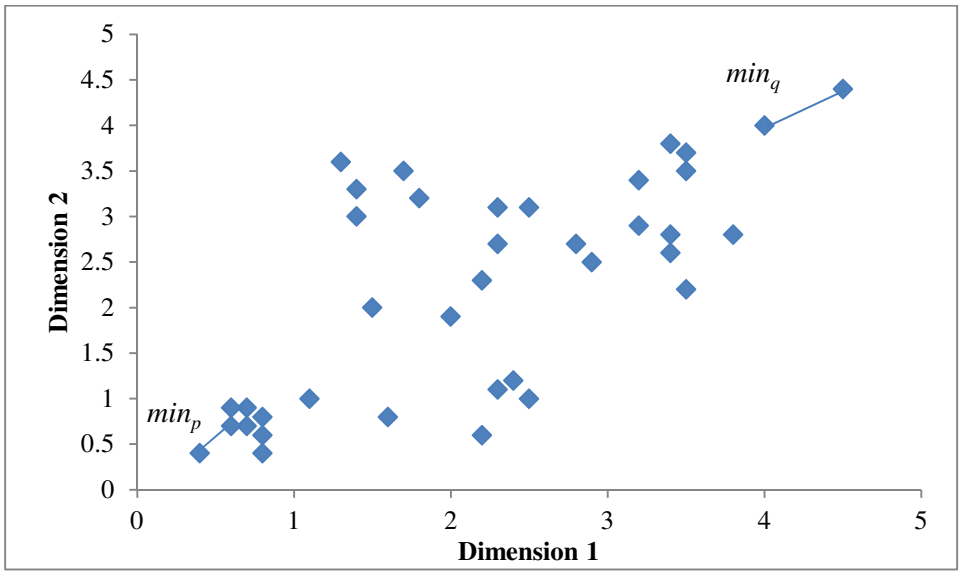


Figure 4.2: Distance of closest data objects to farthest data objects

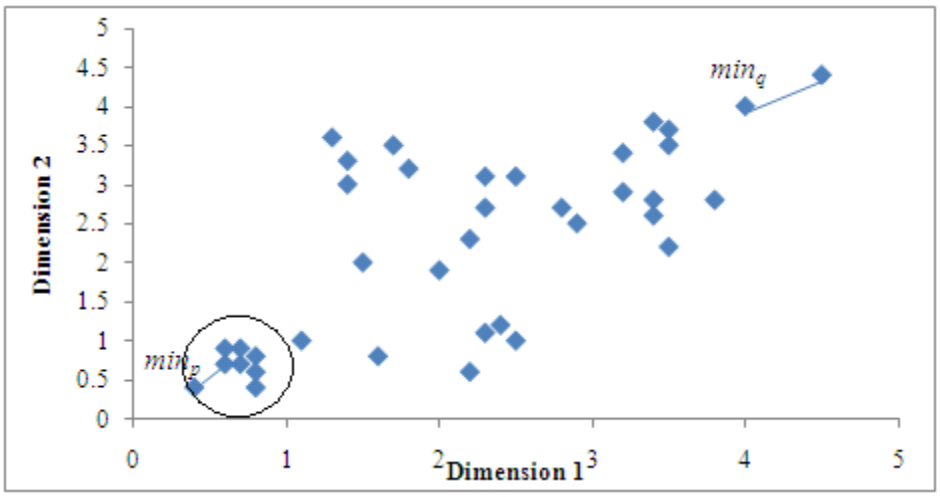


Figure 4.3: Cluster formation with respect to data object *p*

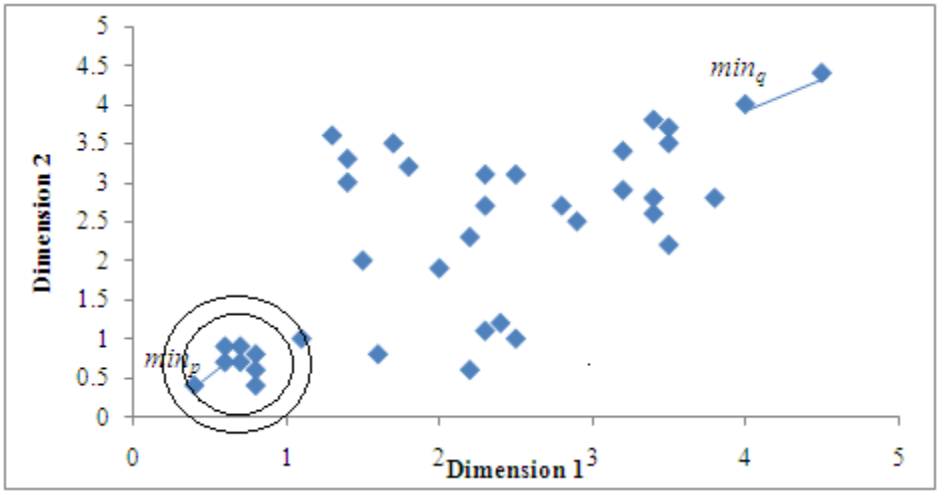


Figure 4.4: A small increment in the neighborhood distance value

4.7 Overlapped/Non-overlapped Clusters and Outlier Detection

In the *ATC* algorithm, overlapped as well as non-overlapped clusters can be generated. During the formation of a cluster, if the clustered data objects are considered un-clustered then these data objects may become its members. In such a case, it generates overlapped clusters; otherwise, non-overlapped clusters are generated. The overlapped data object belongs to more than one cluster, even though its membership may be considered only in one cluster. Membership of overlapped data object depends on the number of data objects in the clusters to which it belongs. In the *ATC* algorithm, overlapped data object is considered as the member of that cluster which has maximum members. It is worth mentioning here that the computation time of *ATC* algorithm increases while generating overlapped clusters.

ATC algorithm also has the ability to detect outliers. As discussed earlier, formation of new cluster starts with the identification of farthest data objects from the un-clustered data objects. So, if a data object is being identified as the farthest data object repeatedly then it may be declared as an outlier.

4.8 Proposed *ATC* Algorithm

Let *p, q* and *r* are three data objects where *p* and *q* are farthest data objects and *r* is closer to *p*. Then, *r* and *p* should be in the same cluster. Closeness of *r* either to *p* or *q* is calculated by using the Euclidean distance measure. It is quite possible that *r* may be at the same distance from *p* and *q.* But, it is not possible that *r* is neither close to *p* nor to *q* because *r* is no farther away than the distance between *p* and *q*; otherwise, *p* and *q* will not be the farthest data objects. If *r* is at the same distance from *p* and *q* then *r* may be clustered with either *p* or *q*. The *ATC* algorithm is given in Figure 4.5. It employs a function *Newcluster*(*r*, *ND*, *k*, *Adj*[*n*][*n*]) to generate clusters. This function generates overlapped clusters and membership of the overlapped data objects as per the process explained in Section 4.7. Size of each cluster generated is compared with the given minimum support value and the small size clusters are considered insignificant clusters. This algorithm always produces the same results on successive runs for a given dataset and minimum support value.

Figure 4.5: Algorithm *ATC* (*D*, *Adj*[*n*][*n*], *min_sup*)

Input: *D* is a set of *n* data objects, *Adj*[*n*][*n*] is an adjacency matrix, *min_sup* is the minimum support value.

Output: Partial dataset partitioned into significant clusters

i. $Let\ \ k = 1$;
ii. $Find\ the\ farthest\ unclustered\ data\ objects\ p\ and\ q\ using\ adjacency\ matrix\ Adj[n][n]$;
iii. $Get\ the\ distance\ of\ closest\ data\ object\ to\ p\ as\ min_p\ and$
$the\ distance\ of\ closest\ data\ object\ to\ q\ as\ min_q\ using\ Adjacency\ matrix\ Adj[n][n]$;
iv. $\boldsymbol{if}\ (min_p > min_q)$
v. $\quad Call\ NewCluster(q, min_p, k, Adj[n][n])$;
vi. $\boldsymbol{else}$
vii. $\quad Call\ NewCluster(p, min_q, k, Adj[n][n])$;
viii. $\boldsymbol{if}\ more\ than\ one\ data\ object\ left\ unclustered$
ix. $\boldsymbol{then}\ k = k + 1$;
x. $\quad \boldsymbol{goto}\ step$ ii.

xi. ***For*** *each overlapped data object*

xii. ***Do*** *assign its membership to that cluster which have maximum number of data objects*;

xiii. ***For*** *each cluster k*

xiv. ***Do if*** $(|k| < min\_sup)$

xv. ***then*** *consider cluster k as insignificant cluster*;

Function NewCluster (*r*, *ND*, *k*, *Adj*[*n*][*n*])

---

i. *Let, data object r as the member of the* k^{th} *cluster and* $|k| = 1$;

ii. *Find all the data objects closer to r within the Neighborhood Distance* (ND) *using adjacency matrix and assign them as members of* k^{th} *cluster*;

//find out the value of δ to be added in the *ND*

iii. *Find the distance between two closest data objects of* k^{th} *cluster as the value of* δ;

iv. $ND = ND + \delta$;

v. ***if*** $(|k|$ *increases*$)$ ***goto*** *step* ii;

4.9 Computational Complexity of *ATC* Algorithm

An adjacency matrix is used to identify the farthest data objects, closest data objects, value of δ and the members of clusters. It requires $O(n^2)$ time to compute adjacency matrix where *n* is the number of data objects. Thus, the time complexity of the *ATC* algorithm is $O(n^2)$, whereas time complexity of *k*-means algorithm is $O(nkl)$ where *n* is the number of data objects, *k* is the number of clusters and *l* is the number of iterations.

4.10 Performance Evaluation of *ATC* Algorithm

This section is divided into three sub-sections. In the first and second sub-sections, the experiments have been performed on artificial and real datasets, respectively. In the third sub-section, *ATC* algorithm is compared with *k*-means algorithm.

4.10.1 Artificial datasets experiments

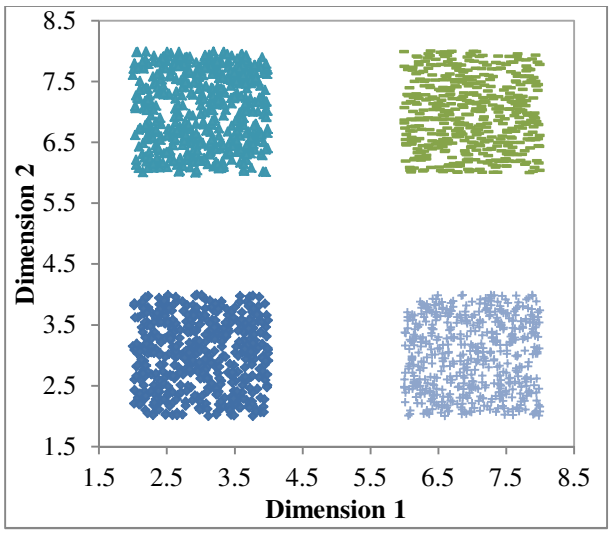
Ten artificially generated datasets have been used to evaluate the performance of *ATC* algorithm. Table 4.1 presents the metadata on these artificial datasets.

A graphical representation of five 2-dimensional artificial uniform datasets AD_2K_1, AD_1.5K_2, AD_1.5K_3, AD_1.5K_4 and AD_1.5K_5 is shown in the Figure 4.6. The graphical representation of non-uniform dataset AD_0.5K_6 is shown in the Figure 4.7.

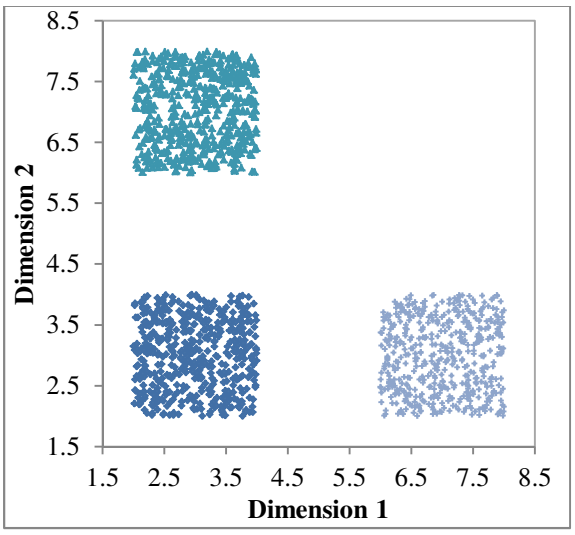
ATC algorithm is executed once on a dataset for a given minimum support value. The clustering results obtained for datasets AD_2K_1, AD_1.5K_2, AD_1.5K_3, AD_1.5K_4, AD_1.5K_5 and AD_0.5K_6 are presented in Figures 4.8 - 4.13, respectively. In these figures, it has been observed that clusters of different sizes have been generated.

Table 4.1: Metadata of artificial datasets

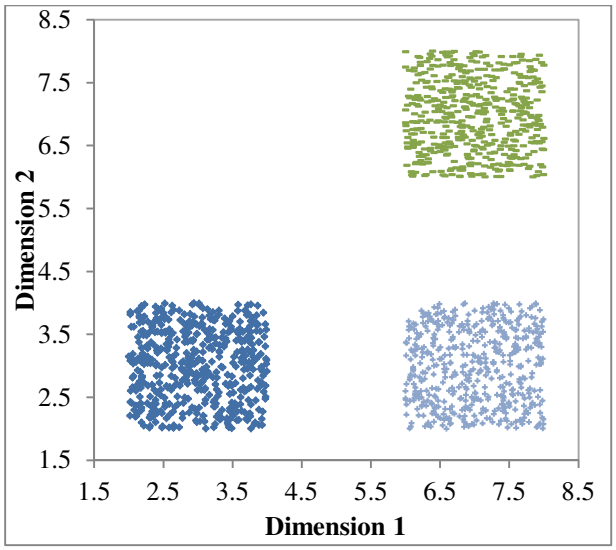
S. No.	Dataset	# Data objects	# Dimension	Distribution of data along axes
1.	AD_2K_1	2000	2	Uniform
2.	AD_1.5K_2	1500	2	Uniform
3.	AD_1.5K_3	1500	2	Uniform
4.	AD_1.5K_4	1500	2	Uniform
5.	AD_1.5K_5	1500	2	Uniform
6.	AD_0.5K_6	500	2	Non-uniform
7.	AD_0.2K_7	200	3	Uniform
8.	AD_80_8	80	3	Non-uniform
9.	AD_0.5K_9	500	10	Uniform
10.	AD_0.5K_10	500	10	Uniform



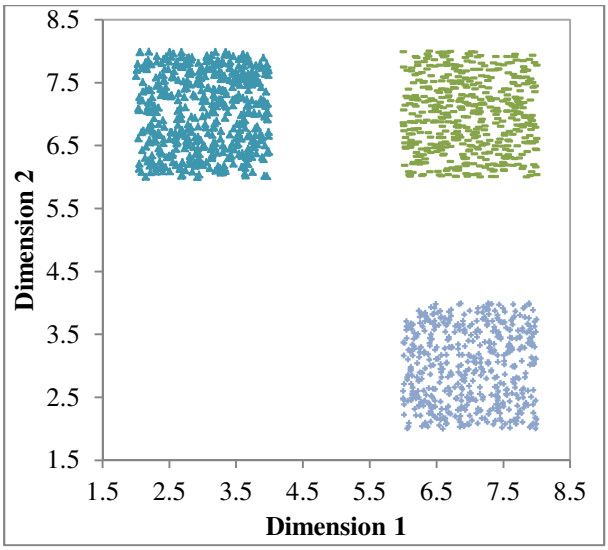
(a)



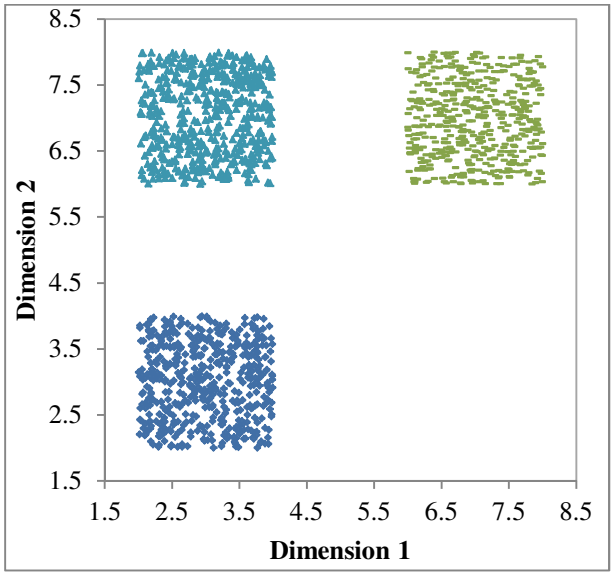
(b)



(c)



(d)



(e)

Figure 4.6: Five artificial datasets (a) AD_2K_1 (b) AD_1.5K_2 (c)AD_1.5K_3 (d) AD_1.5K_4 (e) AD_1.5K_5

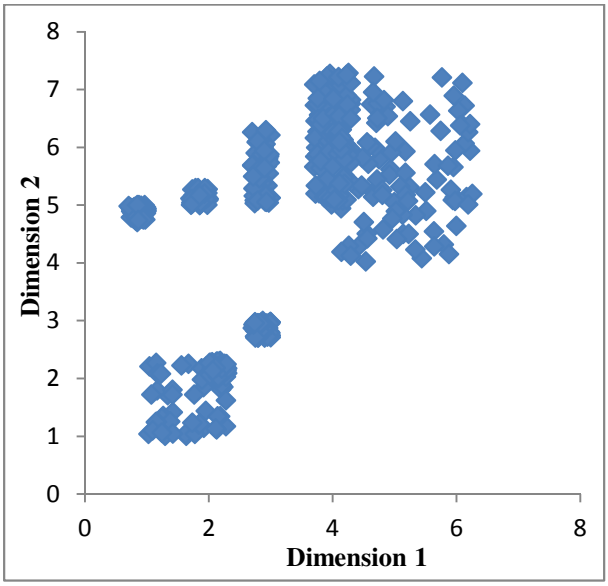


Figure 4.7: AD_0.5K_6 artificial dataset

Clustering results obtained from these artificial datasets are summarized in Table 4.2.

4.10.2 Real datasets experiments

Eight real datasets: *B*reast *C*ancer *W*isconsi-*O*riginal (*BCW-O*), Ecoli, *G*lass *I*dentification (*GI*), Haberman's Survival, Iris, Seed, Wine and Yeast have been used to assess the performance of *ATC* algorithm. These datasets are taken from the UCI repository.

For these datasets, value of minimum support is considered 5% of the size of dataset. The clustering results obtained on the above mentioned datasets are shown in Table 4.3. This table depicts that the significant clusters cover more than 82% of the dataset. To establish the efficiency of *ATC* algorithm, it is further compared with the *k*-means algorithm on artificial and real datasets in the next section.

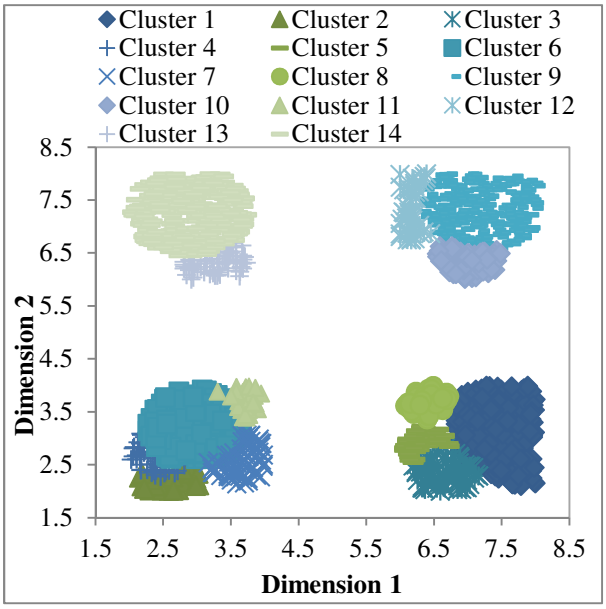


Figure 4.8: Clustered AD_2K_1 artificial dataset (*ATC* algorithm)

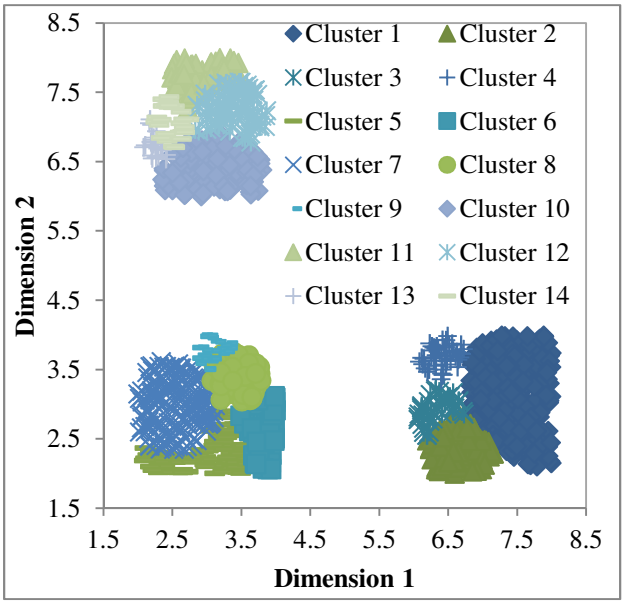


Figure 4.9: Clustered AD_1.5K_2 artificial dataset (*ATC* algorithm)

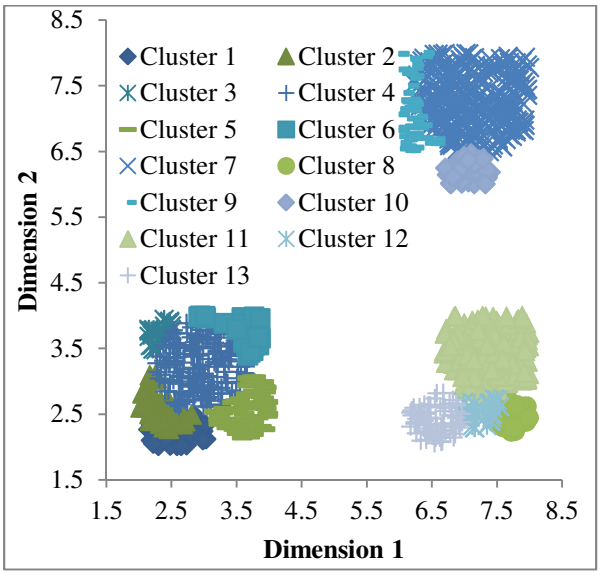


Figure 4.10: Clustered AD_1.5K_3 artificial dataset (*ATC* algorithm)

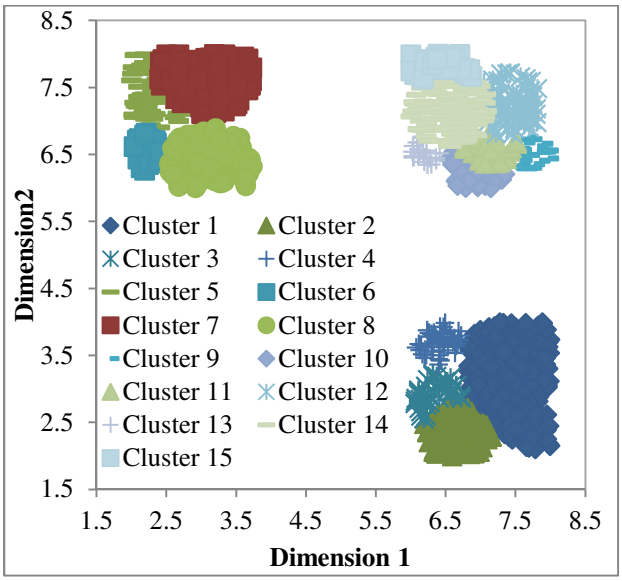


Figure 4.11: Clustered AD_1.5K_4 artificial dataset (*ATC* algorithm)

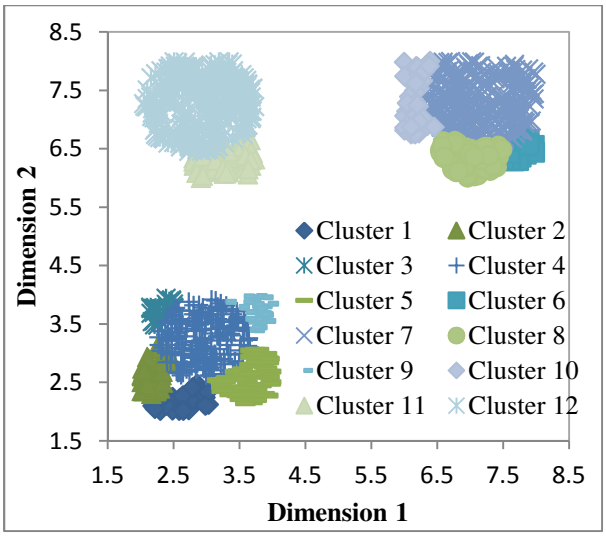


Figure 4.12: Clustered AD_1.5K_5 artificial dataset (*ATC* algorithm)

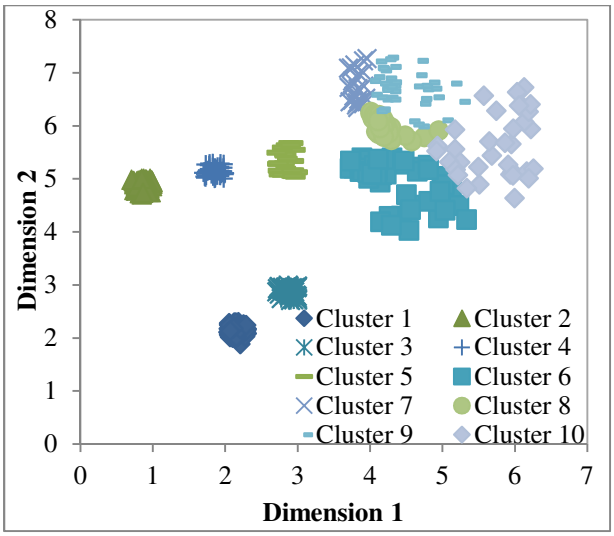


Figure 4.13: Clustered AD_0.5K_6 artificial dataset (*ATC* algorithm)

Table 4.2: Clustering results on artificial datasets using *ATC* algorithm

Dataset	# Clusters identified	Value of minimum support (*min_sup*)	# Significant clusters	# Data objects in significant clusters	Total data objects covered
AD_2K_1	34	1.5%, *i.e.*, minimum 30 data objects/cluster	14	249, 36, 88, 47, 40, 202, 68, 48, 270, 61, 33, 50, 66 and 286	1544, *i.e.*, 77.2% of dataset
AD_1.5K_2	29	1.5%, *i.e.*, minimum 23 data objects/cluster	14	249, 88, 40, 48, 100, 53, 171, 64, 28, 146, 79, 94, 22 and 28	1210, *i.e.*, 80.6 % of dataset
AD_1.5K_3	31	1.5%, *i.e.*, minimum 23 data objects/cluster	13	36, 47, 29, 202, 68, 33, 296, 31, 60, 41, 181, 29 and 72	1125, *i.e.*, 75.0% of dataset
AD_1.5K_4	33	1.5%, *i.e.*, minimum 23 data objects/cluster	15	249, 88, 40, 48, 58, 28, 162, 124, 29, 41, 41, 78, 24, 131 and 37	1178, *i.e.*, 78.5% of dataset

Contd …

AD_1.5K_5	34	1.5%, *i.e.*, minimum 23 data objects/cluster	12	46,57, 39, 212, 78, 35, 280, 71,43, 60, 76 and 296	1093, *i.e.*,72.8 % of dataset
AD_0.5K_6	16	4%, *i.e.*, minimum 20 data objects/cluster	10	51, 50, 45, 49, 25, 46, 22, 28, 41 and 34	391, *i.e.*, 78.2 % of dataset
AD_0.2K_7	18	5%, *i.e.*, minimum 10 data objects/cluster	11	10, 14, 17, 10, 10, 13, 10, 10, 11, 25, and 16	146, *i.e.*, 73.0% of dataset
AD_80_8	5	5%, *i.e.*, minimum 4 data objects/cluster	3	11, 25 and 39	75, *i.e.*, 93.7% of dataset
AD_0.5K_9	8	5%, *i.e.*, minimum 25 data objects/cluster	4	25, 52, 210 and 134	421, *i.e.*, 84.2% of dataset
AD_0.5K_10	7	5%, *i.e.*, minimum 25 data objects/cluster	5	136, 226, 27, 33 and 58	480, *i.e.*, 88.0% of dataset

Table 4.3: Clustering results on real datasets using *ATC* algorithm

Dataset	# Clusters identified	Value of minimum support (*min_sup*)	# Significant clusters	# Data objects in significant clusters	Total data objects covered
BCW-O	7	5%, *i.e.*, minimum 34 data objects/cluster	4	40, 449, 89 and 39	617, *i.e.*, 90.3 % of dataset
Ecoli	7	5%, *i.e.*, minimum 15 data objects/cluster	2	55 and 221	276, *i.e.*, 82.1 % of dataset
GI	5	5%, *i.e.*, minimum 10 data objects/cluster	3	18, 174 and 11	203, *i.e.*, 94.8 % of dataset

Contd …

Haberman	6	5%, *i.e.*, minimum 15 data objects/cluster	4	20, 200, 31 and 44	295, *i.e.*, 96.4 % of dataset
Iris	4	5%, *i.e.*, minimum 7 data objects/cluster	3	31, 96 and 22	149, *i.e.*, 99.3% of dataset
Seed	11	5%, *i.e.*, minimum 10 data objects/cluster	7	26, 21, 38, 20, 25, 21, and 23	174, *i.e.*,82.8% of dataset
Wine	7	5%, *i.e.*, minimum 8 data objects/cluster	4	20, 78, 32 and 29	159, *i.e.*, 95.5% of dataset
Yeast	9	5%, *i.e.*, minimum 74 data objects/cluster	5	150, 718, 81, 143 and 161	1253, *i.e.*, 84.4 % of dataset

4.10.3 Comparison of *ATC* algorithm with *k*-means algorithm

To compare clustering results of *ATC* algorithm on artificial and real datasets, the portion of dataset which lies within the ambit of significant clusters is taken as an input to the *k*-means algorithm. The value of *k* required in the *k*-means algorithm is taken from the dataset clustered using *ATC* algorithm. It is equal to the number of significant clusters generated. In this section, the results of this comparison are presented.

Clustering results of *k*-means algorithm on artificial datasets AD_2K_1, AD_1.5K_2, AD_1.5K_3, AD_1.5K_4, AD_1.5K_5 and AD_1.5K_6 are shown in Figures 4.14 - 4.19, respectively. From these figures, it is evident that different partitions of a given dataset are generated. Clustering results, including comparison between *k*-means algorithm and *ATC* algorithm are summarized in Table 4.4 for these datasets.

k-means algorithm is executed for the same eight real datasets as used by the *ATC* algorithm. The clustering results obtained for these datasets are presented in Table 4.5 and it has been observed that *k*-means algorithm partitions the dataset into *k* clusters such that the range of the number of data objects in a cluster is narrow (bold faced values in column 4 of Table 4.4 and Table 4.5) whereas the proposed algorithm can handle a wider range (bold faced values

in column 5 of Table 4.4 and Table 4.5). It indicates that *k*-means algorithm generates clusters of almost same sizes whereas *ATC* algorithm generates clusters of different sizes.

4.11 Key Observations, Advantages and Limitations of *ATC* Algorithm

We have noted the following observations from the experiments conducted in the work presented in this chapter.

- *ATC* algorithm automatically clusters the datasets without knowing the number of clusters.
- A large number of clusters are generated using this algorithm. However, the parameter minimum support plays an important role to obtain significant clusters.
- Significant clusters cover almost all the data objects of a given dataset.
- The proposed algorithm is capable of generating overlapped and non-overlapped clusters.
- Since the algorithm is deterministic in nature, it gives same result on successive runs for a given dataset and a given minimum support value.
- The order of selection of data objects does not affect the clustering results because the proposed algorithm does not require any user specified parameter to generate clusters. However, user specified parameter, minimum support, is used to improve the clustering efforts but does not play any role in the formation of clusters.

ATC algorithm has the following advantages:

- It is deterministic algorithm.
- It automatically generates the clusters.
- It can detect outliers.
- It can generate overlapped and non-overlapped clusters.

Limitations of *ATC* algorithm are:

- It does not generate non-spherical clusters as it is based on centroid based clustering.

- It requires adjacency matrix as the prerequisite, which increases the time complexity of the algorithm.

Therefore, the proposed *ATC* algorithm can be used as an alternate clustering algorithm.

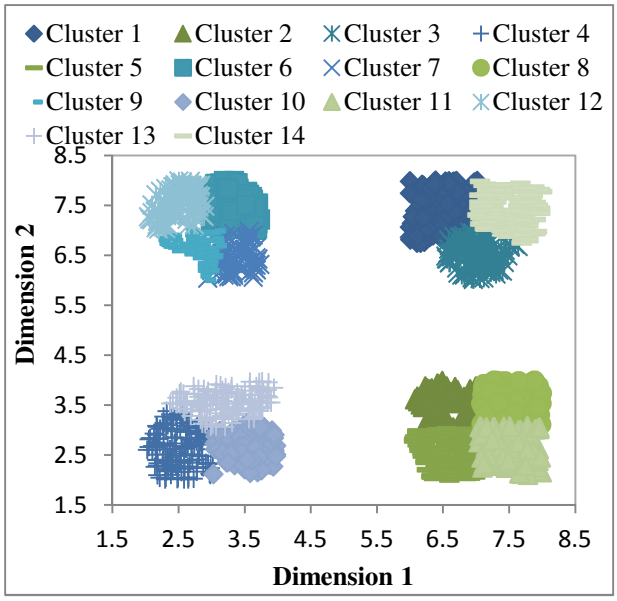


Figure 4.14: Clustered AD_2K_1 artificial dataset (*k*-means algorithm)

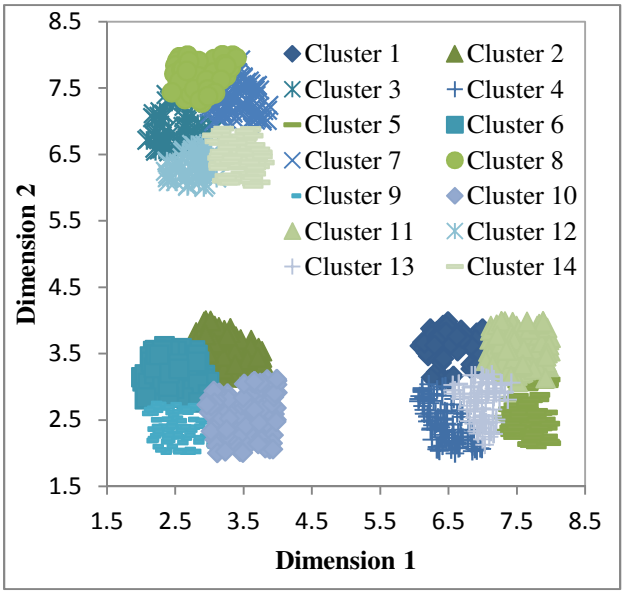


Figure 4.15: Clustered AD_1.5K_2 artificial dataset (*k*-means algorithm)

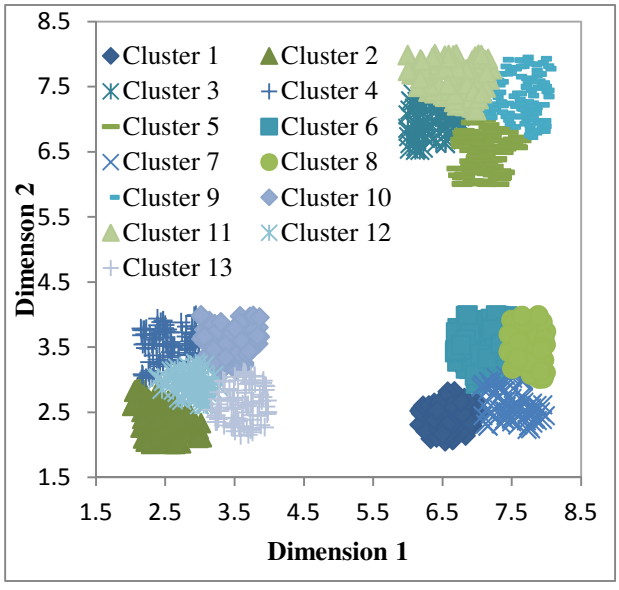


Figure 4.16: Clustered AD_1.5K_3 artificial dataset (*k*-means algorithm)

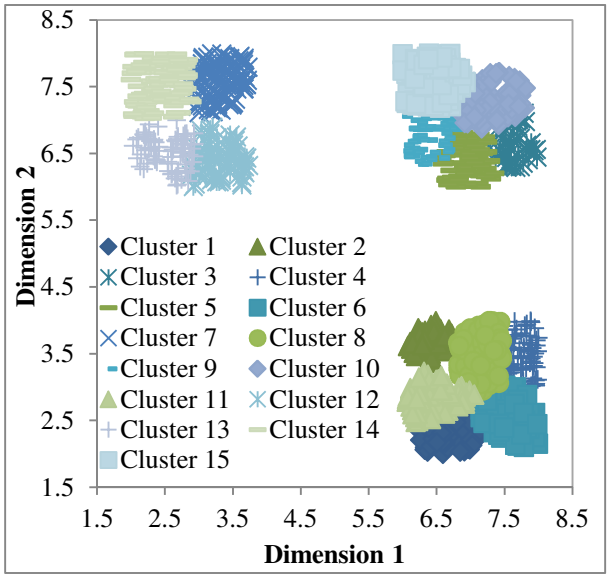


Figure 4.17: Clustered AD_1.5K_4 artificial dataset (*k*-means algorithm)

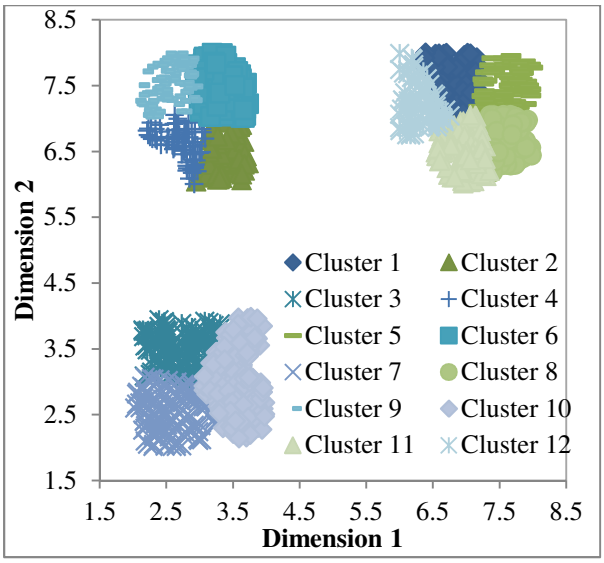


Figure 4.18: Clustered AD_1.5K_5 artificial dataset (*k*-means algorithm)

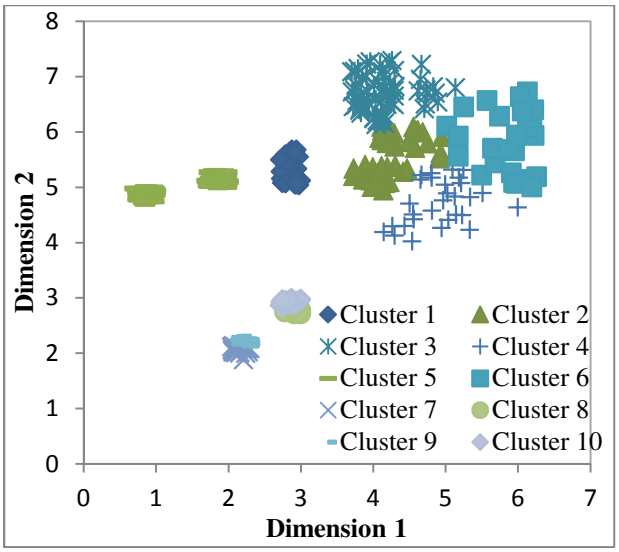


Figure 4.19: Clustered AD_0.5K_6 artificial dataset (*k*-means algorithm)

Table 4.4: Comparison of *k*-means algorithm and *ATC* algorithm on artificial datasets

Dataset	# Clusters	Total data objects	*k*-means algorithm	*ATC* algorithm
			# Data objects in Clusters	# Data objects in Clusters
AD_2K_1	14	1544	**151**, 83, 108, 145, 125, 117, **71**, 114, 77, 102, 103, 87, 139 and 122	249, 36, 88, 47, 40, 202, 68, 48, 270, 61, **33**, 50, 66 and **286**
AD_1.5K_2	14	1210	73, 97, **68**, 89, 75, 106, 87, 78, 83, **130**, 111, 62, 77 and 74	**249**, 88, 40, 48, 100, 53, 171, 64, 28, 146, 79, 94, **22** and 28
AD_1.5K_3	13	1125	79, 84, 80, 93, 95, 91, 73, 70, **116**, 85, 106, **69** and 84	36, 47, 29, 202, 68, 33, **296**, 31, 60, 41, 181, **29** and 72
AD_1.5K_4	15	1178	78, **48**, 49, 68, 79, 78, **115**, 88, 73, 90, 65, 78, 77, 102 and 90	**249**, 88, 40, 48, 58, 28, 162, 124, 29, 41, 41, 78, **24**, 131 and 37

Contd …

AD_1.5K_5	12	1093	103, 81, 139, 87, 85, 127, 142, **73**, 97, **164**, 106 and 89	46,57, **39**, 212, 78, 35, 280, 71,43, 60, 76 and **296**
AD_0.5K_6	10	391	25, 50, 65, 29, **99**, 27, 25, **17**, 26 and 28	**51**, 50, 45, 49, **25**, 46, 22, 28, 41 and 34
AD_0.2K_7	11	146	8, **26**, 7, 11, 15, 17, 11, 23, 13, **5** and 10	10, 14, 17, 10, 10, 13, 10, **10**, 11, **25**, and 16
AD_80_3_8	3	75	**36**, **18** and 21	**11**, 25 and **39**
AD_0.5K_9	4	421	109, 96, **91**and **125**	**25**, 52, **210** and 134
AD_0.5K_10	5	480	114, 90, **122**, 92 and **62**	136, **226**, **27**, 33 and 58

Table 4.5: Comparison of *k*-means algorithm and *ATC* algorithm on real datasets

Dataset	# Clusters (*k*)	Total data objects	*k*-means algorithm	*ATC* algorithm
			# Data objects in clusters	# Data objects in clusters
BCW-O	4	617	179, **79**, 94 and **265**	40, **449**, 89 and **39**
Ecoli	2	276	**204** and **72**	**55** and **221**
GI	3	203	89, **23** and **91**	18, **174** and **11**
Haberman	4	295	**31**, 74, 81 and **109**	**20**, **200**, 31 and 44
Iris	3	149	49, **39** and **61**	**31**, **96** and 22
Seed	7	174	25, **18**, 24, 26,27, 25 and **29**	26, 21, **38**, **20**, 25, 21, and 23
Wine	4	159	**70**, 56, 25 and **8**	**20**, **78**, 32 and 29
Yeast	5	1253	341, 214, **124**, 208 and **366**	150, **718**, **81**, 143 and 161

Chapter Summary

In this chapter, a deterministic partitioning based algorithm, *A*daptive *T*hreshold based *C*lustering (*ATC*) algorithm has been proposed. The proposed algorithm uses a parameter, neighborhood distance, to cluster the data objects. Neighborhood distance is not specified by the user, rather, it is calculated automatically and moreover, it is an adaptive parameter. Another parameter used in *ATC* algorithm is the minimum support value which prunes the insignificant clusters. Performance of the *ATC* algorithm is also assessed on artificial and real datasets in this chapter and observed that it is capable of detecting outliers and generates overlapped and non-overlapped clusters. It has an advantage over the single pass and modified single pass clustering algorithms as it does not depend on the order of the selection of the data objects. Here, it has also been observed from the experiments that *ATC* algorithm generates clusters of different sizes whereas *k*-means algorithm generates clusters of almost same sizes.

Chapter 5

New Cluster Validation Measures

Clustering algorithms are highly sensitive to the characteristics of dataset especially noise and dimensions. Most of them depend on initial parameters and assumptions, in order to partition the dataset into clusters. Therefore, they exhibit different results during successive runs. So, it is necessary to validate the results of clustering algorithms to apply them in various applications. Validation is a process of measuring the quality of clustering algorithms. In this chapter, new validation measures are proposed for estimating how well a partition fits into a structure underlying the data. An experiment has been performed to justify that proposed measures can be appropriate for the validation purposes.

5.1 Validity Measures

Cluster validation is quite difficult and a challenging task in the field of clustering. Cluster validation techniques are divided into two groups: External and Internal. External validation technique relies on the available information regarding correct partition whereas internal validation technique relies on the structure in the dataset. Thus, in the case of non-availability of external information, internal validation is the widely acceptable option. Internal validation technique is defined by two validity measures: compactness and separation.

5.2 Existing Compactness Measures

Compactness is the measure of intra-cluster distance of the data objects in a cluster which should be minimum. Alternatively, it is the measure of maximum similarity of all data objects in a cluster.

For measurement of compactness, the following two popular measures are available:

- Centroid Linkage

  In this measure, compactness is computed by averaging the distance of data objects of a cluster to the centroid of that cluster. This average distance value is also known as variance of the cluster. Variance of each cluster is averaged to get overall compactness of the partition obtained for a given dataset. Low value of compactness indicates the better partitioning.

- Averaged paired distance

  In this measure, instead of calculating the distance of data objects to the centroid, distance between pair of data objects is computed.

Details of these measures are given in Section 1.5.

5.3 Proposed Compactness Measures

In this section, new methods to measure the compactness of a cluster are proposed. In these measures, for each dimension, mean and standard deviation of data objects are calculated. These are used to calculate the dispersion of data objects along the dimensions. Maximum dispersed value in each dimension is averaged to know the compactness of a cluster. There are two ways to take the average of maximum dispersion value along the dimensions.

5.3.1 Arithmetic dispersion

In this measure, for a cluster *C*, compactness is defined by considering μ^i as the mean and σ^i as the standard deviation of data objects along the i^{th} $(1 \leq i \leq d)$ dimension. For the data object $\boldsymbol{d_j}$ along its i^{th} dimension, it can be stated that:

$$\mu^i - c\sigma^i \leq d_j^i \leq \mu^i + c\sigma^i \quad \ldots (5)$$

where c is a constant. From (5), it can be inferred that

$$-c\sigma^i \leq d_j^i - \mu^i$$

$$\Rightarrow -c \leq \frac{d_j^i - \mu^i}{\sigma^i} \quad \ldots (6)$$

Now, from the same equation it can also be inferred that

$$c\sigma^i \geq d_j^i - \mu^i$$

$$\Rightarrow c \geq \frac{d_j^i - \mu^i}{\sigma^i} \quad \ldots (7)$$

Now, from (6) and (7) the following relation can be derived:

$$-c \leq \frac{d_j^i - \mu^i}{\sigma^i} \leq c$$

The large value of $\frac{d_j^i - \mu^i}{\sigma^i}$ depicts that $\boldsymbol{d_j}$ is maximum dispersed data object along the i^{th} dimension in cluster *C*. Now, maximum dispersed value along each dimension is averaged arithmetically to know the compactness of cluster C. This proposed compactness measure is named as *A*rithmetic *D*ispersion (*AD*). It is mathematically defined as:

$$Comp_{AD}(C) = \frac{1}{d}\sum_{k=1}^{d} c^k$$

where $c^k = max_{1 \leq j \leq n} \frac{d_j^i - \mu^i}{\sigma^i}$. Value of $Comp_{AD}(C)$ for each cluster is arithmetically averaged to get the overall compactness of the partition obtained for a given dataset.

5.3.2 Geometric dispersion

In this section, another method to measure the compactness is proposed. In this method, maximum dispersion values along each dimension, as obtained in Section 5.3.1 are averaged geometrically to calculate the compactness of cluster C. This proposed compactness measure is named as *G*eometric *D*ispersion (*GD*). It is mathematically defined as:

$$Comp_{GD}(C) = \left(\prod_{k=1}^{d} c^k\right)^{1/d}$$

where $c^k = max_{1 \leq j \leq n} \frac{d_j^i - \mu^i}{\sigma^i}$. Value of $Comp_{GD}(C)$ for each cluster is geometrically averaged to get the overall compactness of the partition obtained for a given dataset.

Performance evaluation of these proposed measures has been given in Section 5.6.

5.4 Existing Separation Measures

Separation is the measure of inter-cluster distance between clusters. A higher value of this measure indicates a lower similarity between data objects of clusters.

Separation between clusters C_r and C_s can be measured by the following three existing measures:

- Single linkage: It is the minimum distance between data objects of clusters C_r and C_s.
- Centroid linkage: It is the distance between centroids of clusters C_r and C_s.

- Complete linkage: It is the maximum distance between data objects of clusters C_r and C_s.

5.5 Proposed Separation Measure

In this section, a new separation measure is proposed. In this measure, average distance between data objects of two clusters is calculated by considering all data objects of one cluster and pairing them with all data objects of other cluster. This averaged value is defined as the separation between clusters C_r and C_s. This proposed separation measure is named as *A*verage *L*inkage (*AL*). It is mathematically defined as:

$$Sep_{AL}(C_r, C_s) = \frac{\sum_{x_i^k \in C_r,\ x_j^k \in C_s} \sqrt{\sum_{k=1}^{d} (x_i^k - x_j^k)^2}}{|C_r||C_s|}$$

In the above formula, $|C_r|$ and $|C_s|$ are the number of data objects in clusters C_r and C_s, respectively. Performance evaluation of this proposed measure has been given in Section 5.7.

5.6 Performance Evaluation of Compactness Measures

This section is divided into two sub-sections. In the first sub-section, experiments have been performed on the artificial datasets and in the second sub-section, experiments on real datasets have been performed. Clustering of these datasets is done using *k*-means, *SPC* and *MSPC* algorithms.

5.6.1 Experiments on artificial datasets

To perform the experiments, fifteen artificial datasets used in third chapter have been taken. These data objects are generated randomly in the range of 100 to 499 in both dimensions. In *k*-means algorithm, the value of *k* is taken in the range of 4 to 10, in *SPC* algorithm, the value of threshold is taken in the range of 50 to 75 and in *MSPC* algorithm, threshold similarity value is taken as mean of paired distance of data objects. In these experiments, performance of existing and proposed compactness measures is compared for *k*-means, *SPC* and *MSPC* algorithms. The results of this comparison are depicted in Figures 5.1 - 5.3.

From these figures, it is evident that arithmetic and geometrical dispersion measures provide almost same value to the centroid based compactness measure. For high dimension datasets, these can also be used as potential measures. In these measures, compactness is based on

maximum dispersion of data objects along each dimension. As such, low values of these measures indicate that data objects of a cluster are close to each other.

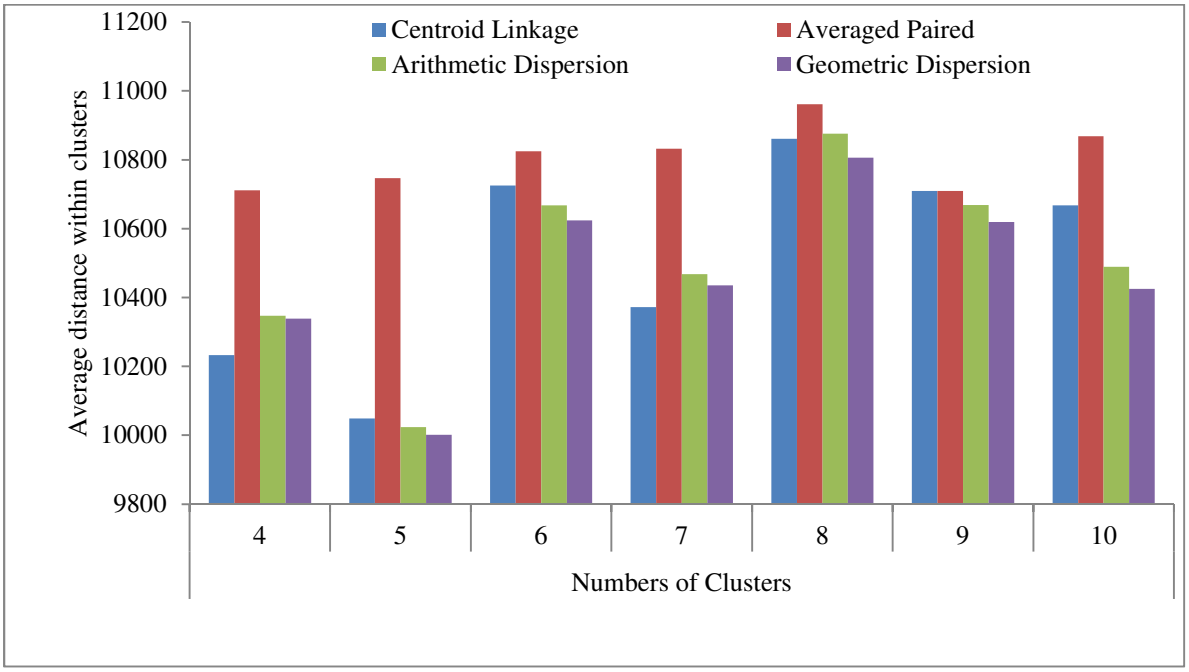
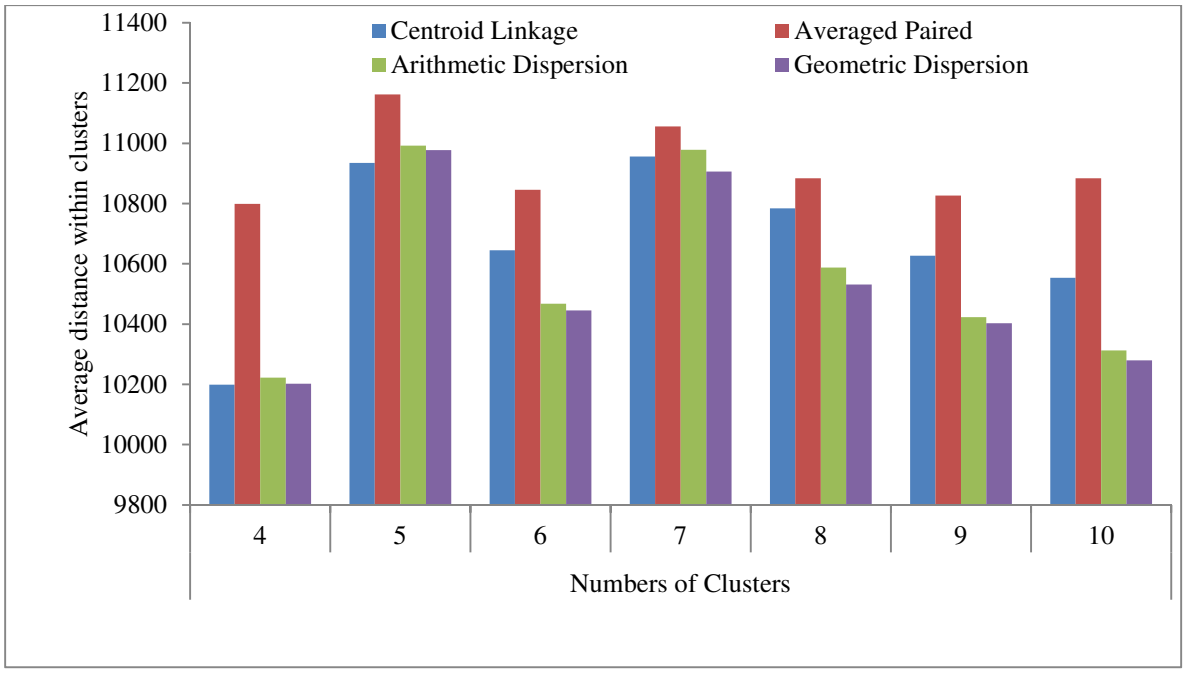


Figure 5.1: Compactness measures using *k*-means algorithm



Figures 5.2: Compactness measures using *SPC* algorithm

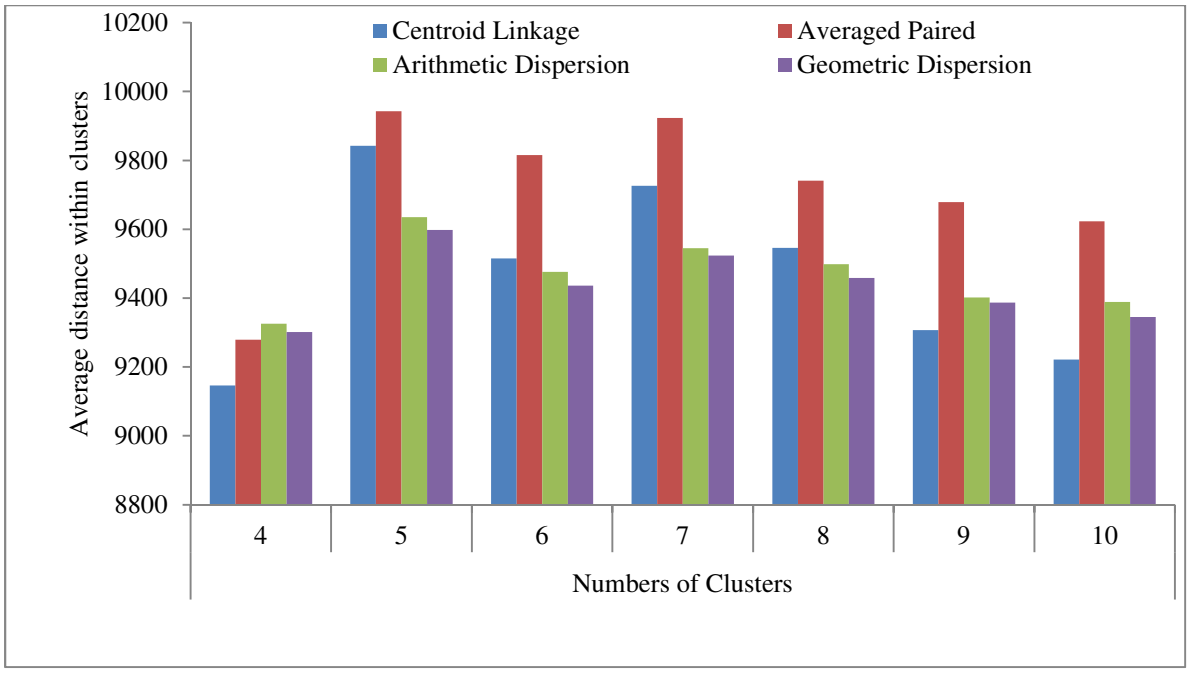


Figure 5.3: Compactness measures using *MSPC* algorithm

5.6.2 Experiments on real datasets

Experiments have been carried out on four real datasets: Iris, Seeds, Ecoli and Wine to analyze the performance of existing and proposed compactness measures. Comparison of these measures for *k*-means, *SPC*, and *MSPC* algorithms is shown in Figures 5.4 - 5.7.

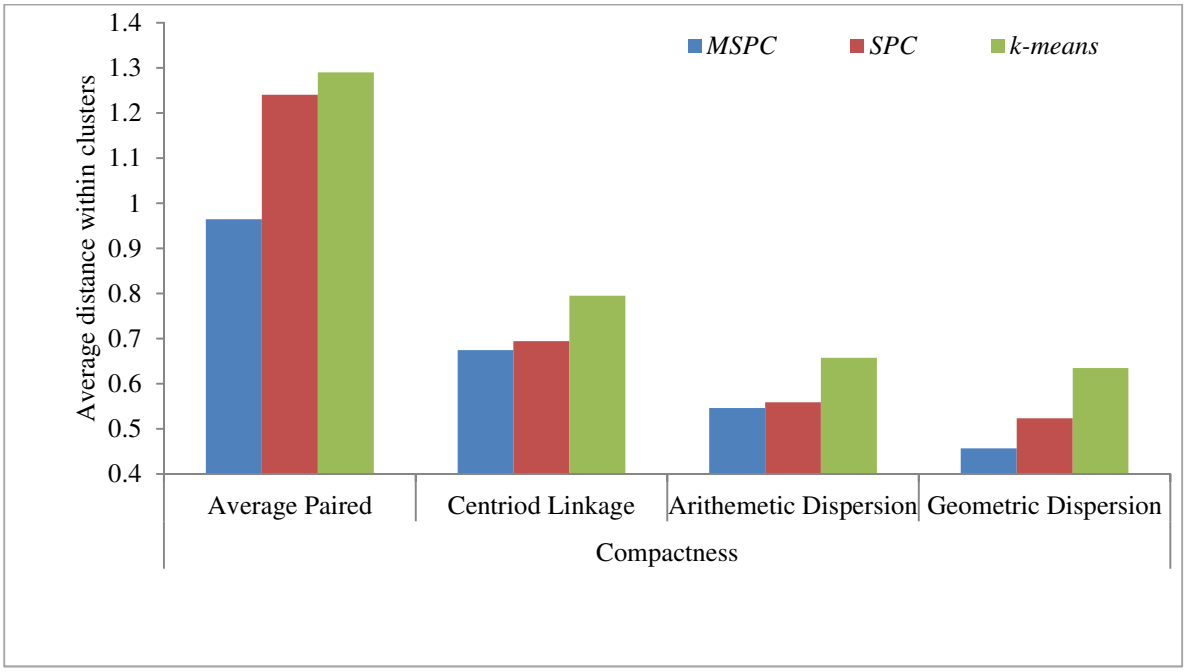


Figure 5.4: Compactness measures for Iris dataset

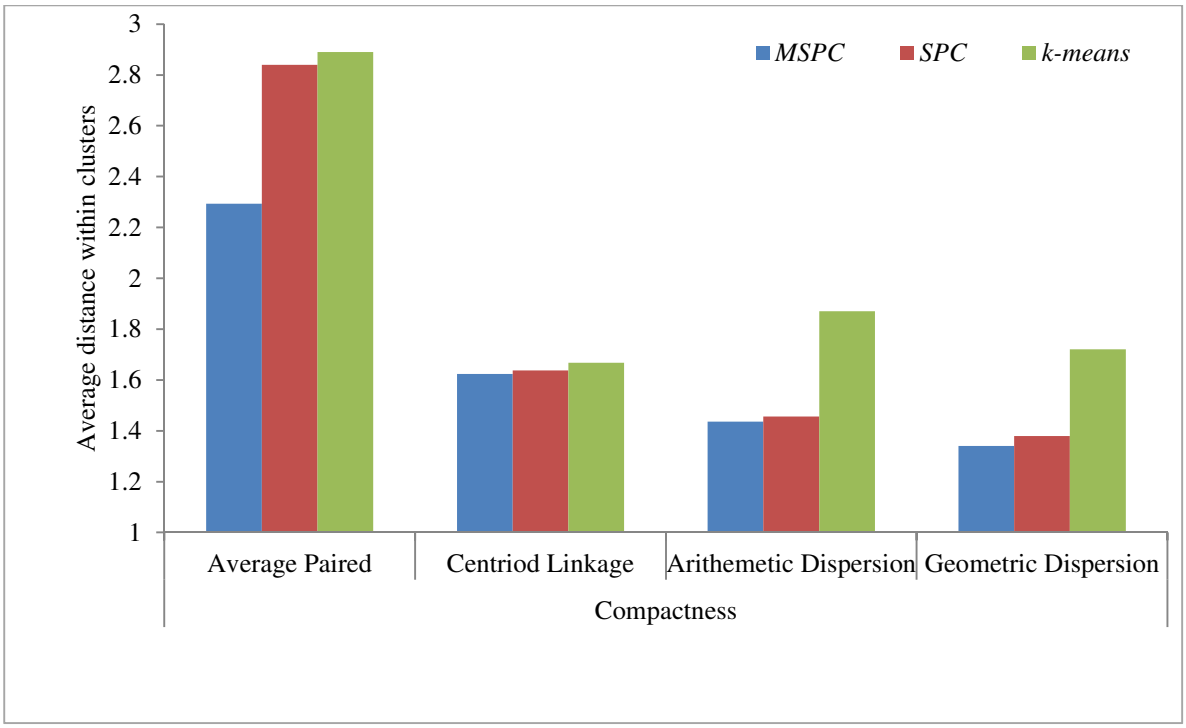


Figure 5.5: Compactness measures for Seeds dataset

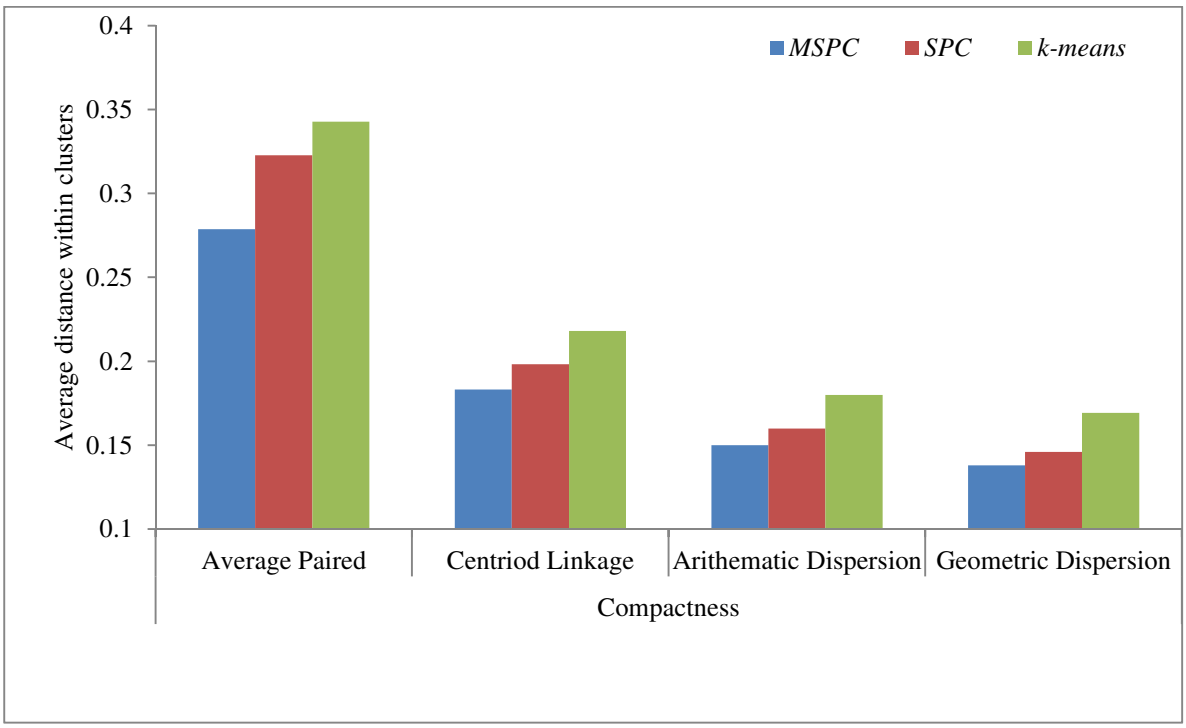


Figure 5.6: Compactness measures for Ecoli dataset

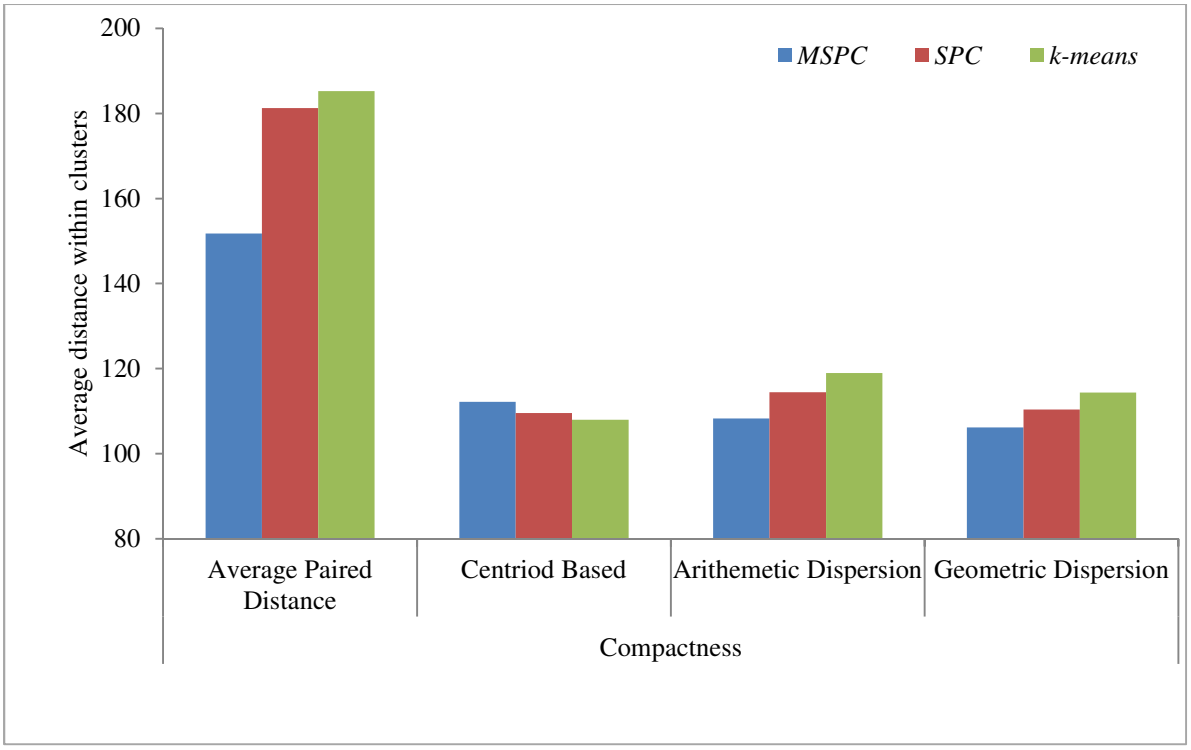


Figure 5.7: Compactness measures for Wine dataset

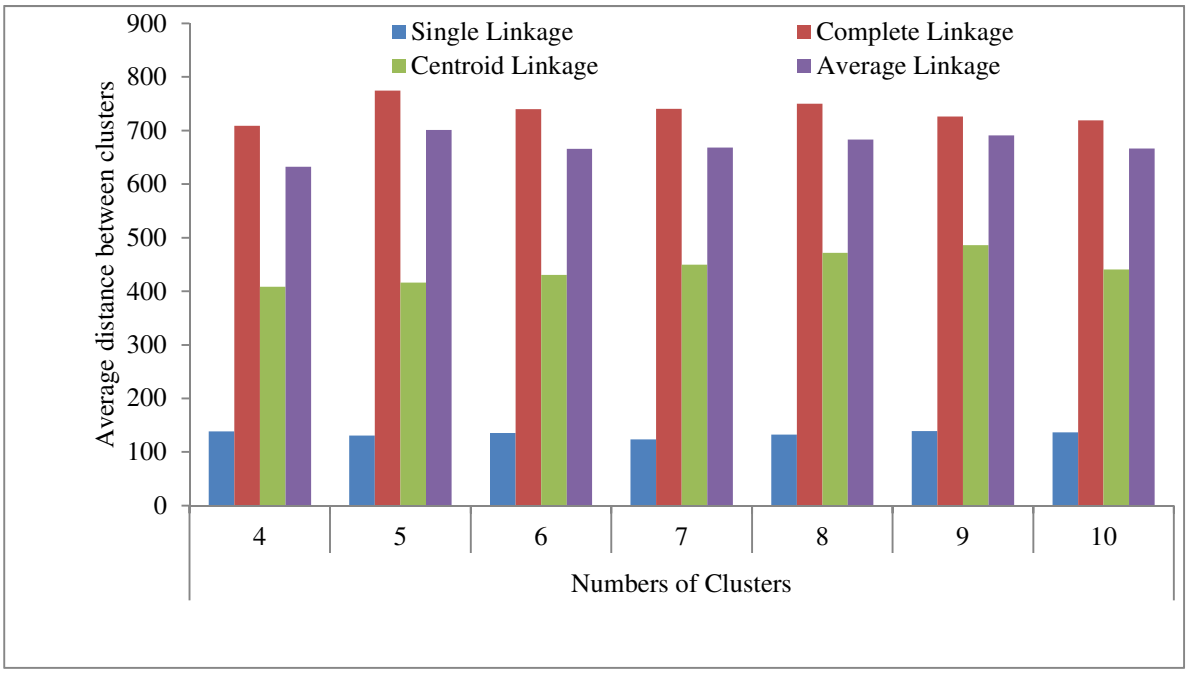
It is evident from these figures that proposed measures can also be act as potential compactness measures. This is worth mentioning here that number of clusters for three algorithms *k*-means, *SPC*, and *MSPC* have been fixed as per the actual number of clusters in these datasets, as given in Table 3.1. We have generated three clusters for Iris, Seeds and Wine datasets and eight clusters for Ecoli dataset. It can also be noted from Figures 5.4 - 5.7 that both, arithmetic and geometric dispersion, compactness measures perform better than existing measures, namely, averaged paired and centroid linkage for Iris, Seeds and Ecoli datasets. In the next section, performance of existing and proposed separation measures is presented.

5.7 Performance Evaluation of Separation Measures

This section is also divided into two sub-sections. In the first sub-section, experiments have been performed on the artificial datasets and in the second sub-section, experiments on real datasets have been performed. Clustering of these datasets is done using *k*-means, *SPC* and *MSPC* algorithms.

5.7.1 Artificial datasets experiments

To perform the experiments, same fifteen artificial datasets have been used. In these experiments, performance of existing and proposed separation measures is compared in Figures 5.8 - 5.10 for *k*-means, *SPC* and *MSPC* algorithms, respectively.



Figures 5.8: Separation measures using *k*-means algorithm

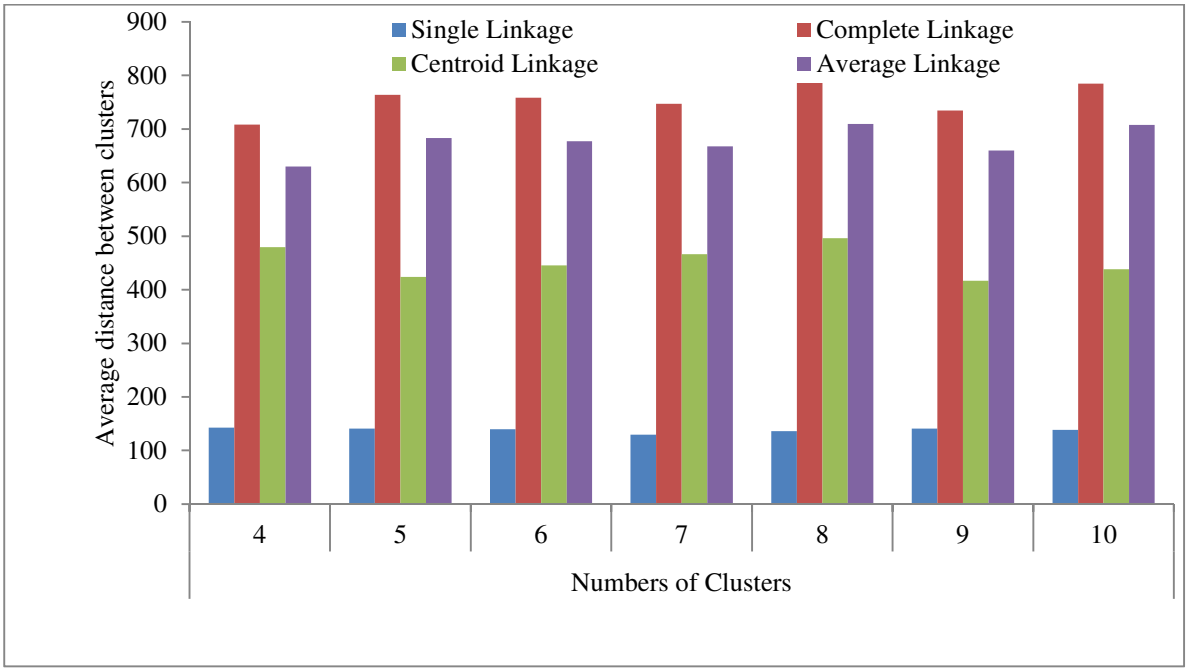


Figure 5.9: Separation measures using *SPC* algorithm

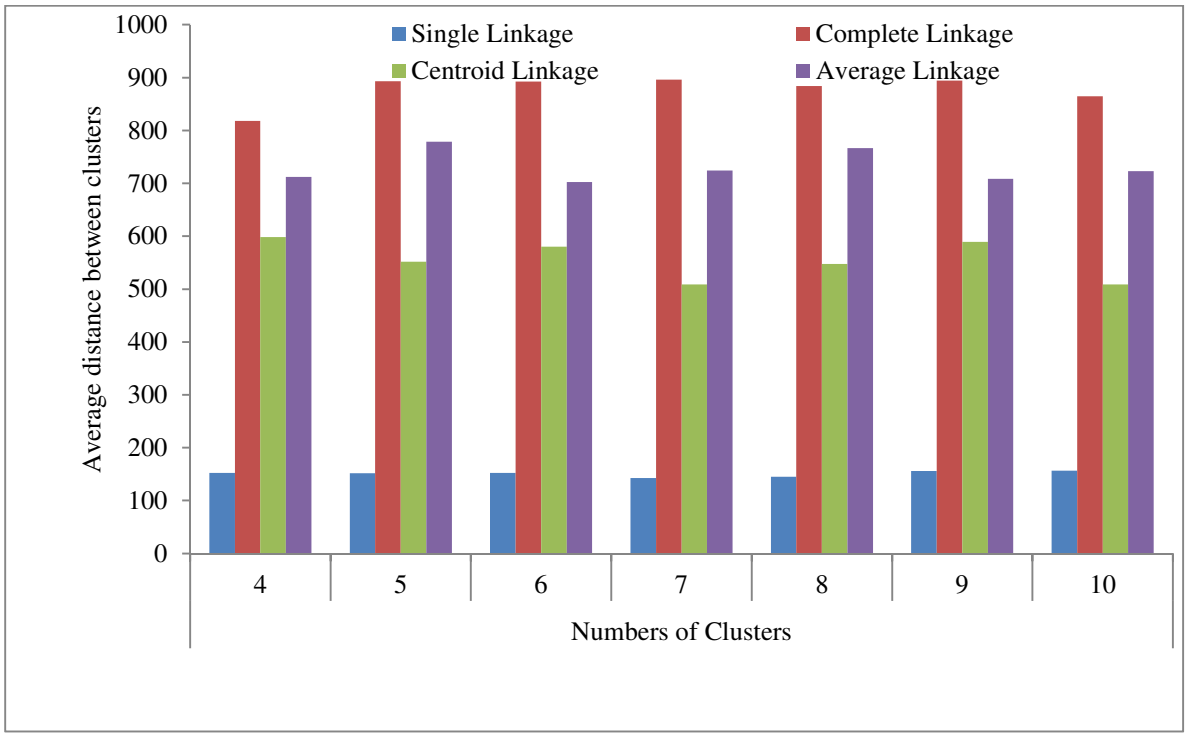


Figure 5.10: Separation measures using *MSPC* algorithm

It is evident from these figures that proposed measures can also be act as potential separation measure. In the next section, performance of existing and proposed separation measures is presented on real datasets.

5.7.2 Real datasets experiments

Four high dimensional real datasets, namely, Iris, Seeds, Ecoli and Wine have again been used in the experiments in this section. To perform these experiment, value of k for k-means algorithm and threshold similarity value (T_{th}) for *SPC* and *MSPC* algorithms is considered the same as in section 5.6.2. A comparison of all separation measures on these datasets for k-means, *SPC* and *MSPC* algorithms is presented in Figures 5.11 - 5.14. From these figures, it is observed that the value of proposed separation measure lies between the centroid and complete linkage separation measures. Average linkage is a more accurate measure due to the involvement of each and every data object as compared to single linkage, centroid linkage and complete linkage separation measures. However, it is computationally more costly.

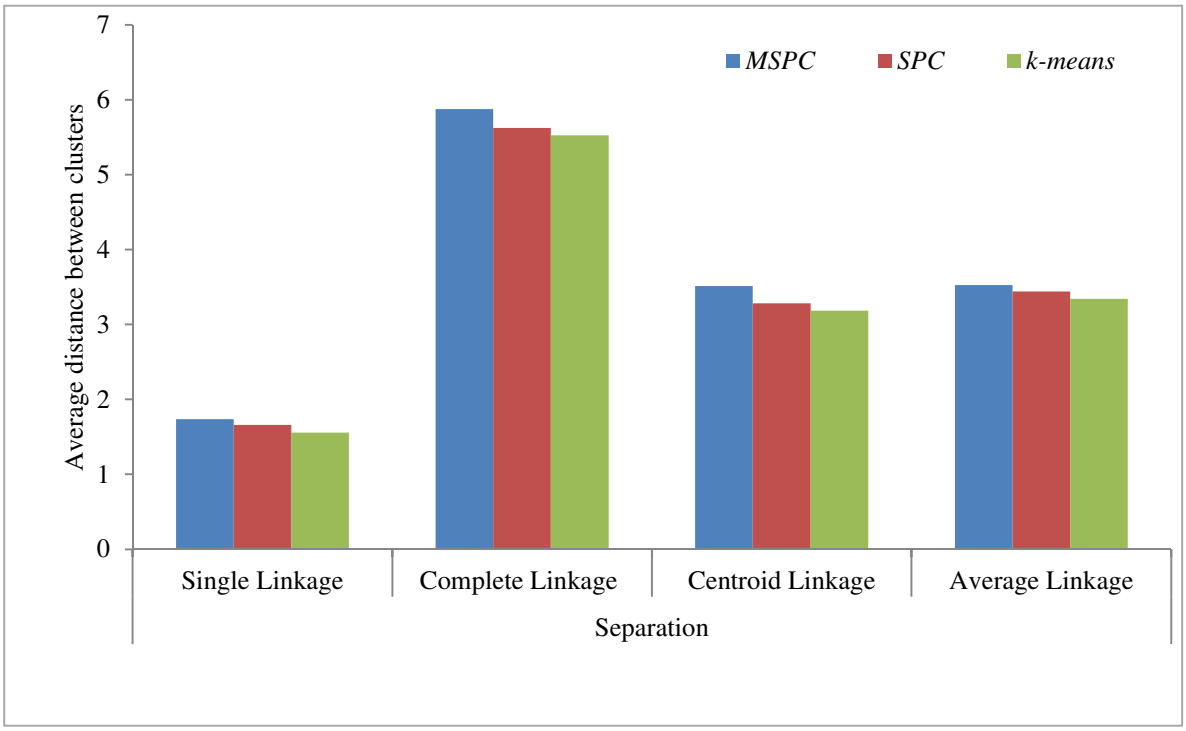


Figure 5.11: Separation measures for Iris dataset

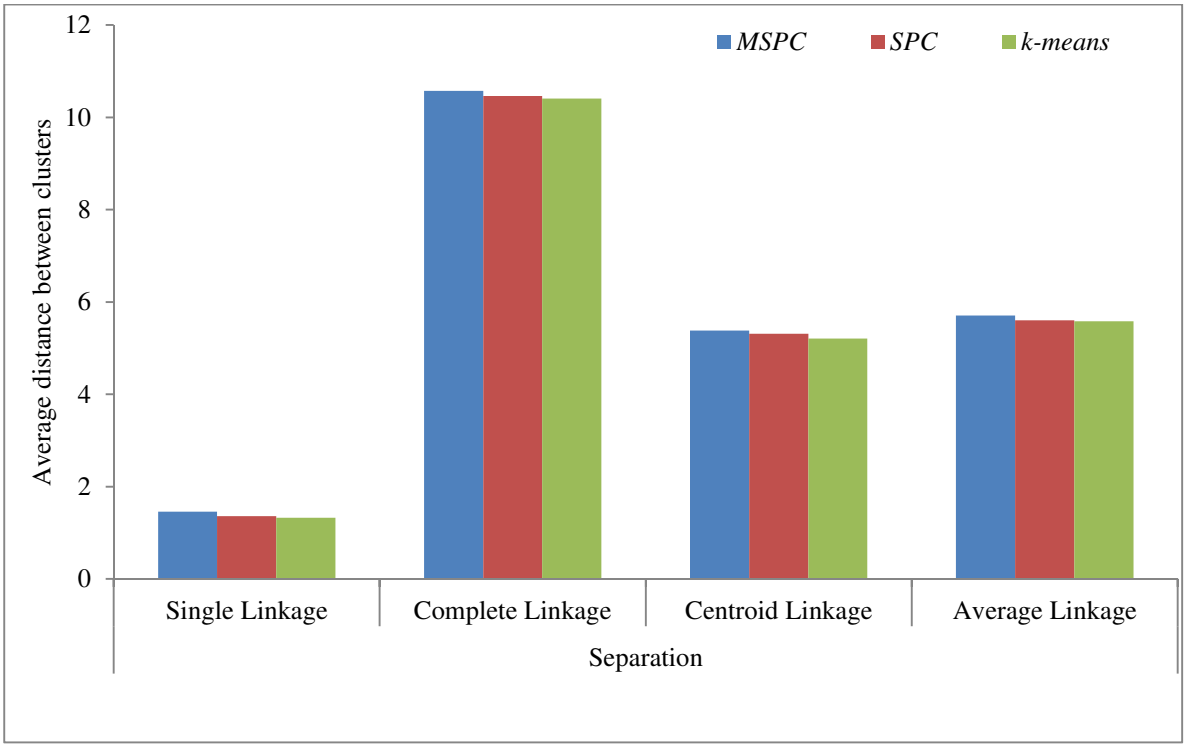


Figure 5.12: Separation measures for Seeds dataset

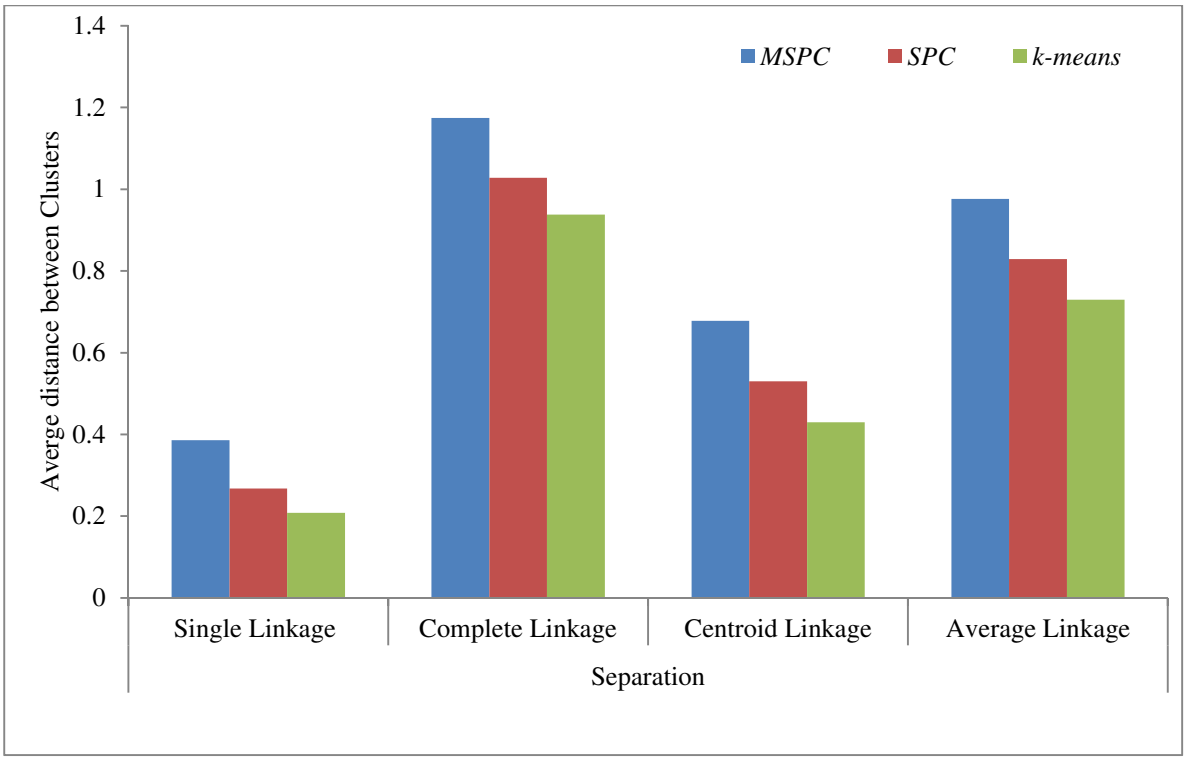


Figure 5.13: Separation measures for Ecoli dataset

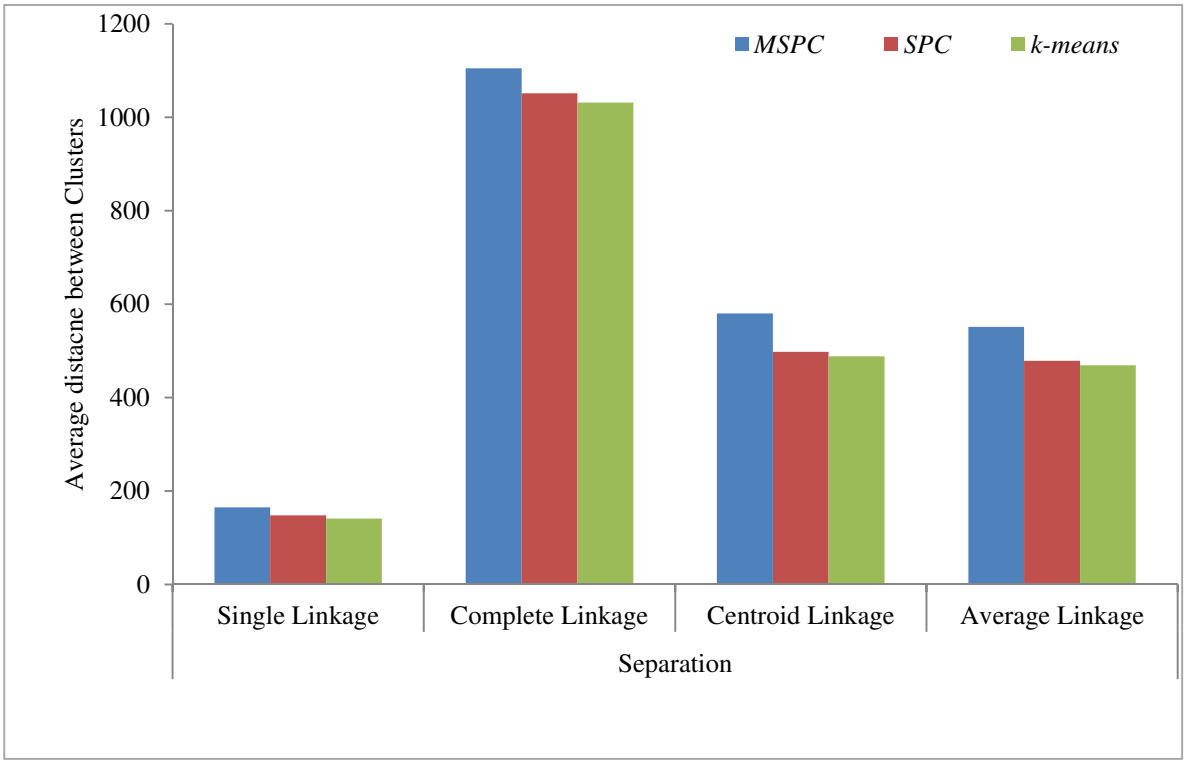


Figure 5.14: Separation measures for Wine dataset

Chapter summary

In this chapter, new separation and compactness measures have been proposed. Proposed compactness measures are based on the dispersion of data objects along each dimension. The maximum dispersion value along each dimension is averaged to get the compactness of a cluster. Proposed separation measure is an averaged paired distance between the data objects of clusters. Experiments have been carried out on artificial and real datasets to justify these measures. We have here claimed that proposed measures can also act as potential compactness and separation measures.

Chapter 6
Conclusion and Future Scope

Clustering is an active research area even these days as this problem is complex in nature. So, there is always a scope of improvement which motivates the researchers to work on this problem. Efficient detection and interpretation of clusters is the main objective of this research work. In order to achieve this objective, two clustering algorithms: modified single pass clustering algorithm and adaptive threshold based clustering algorithm have been proposed. Credibility of these clustering algorithms is validated through validation techniques. New cluster validation measures have also been proposed in this research work.

6.1 Conclusion

In this research work, extensive literature has been explored for various clustering algorithms. We mainly focused on two existing clustering algorithms, namely, *k*-means and *S*ingle *P*ass *C*lustering (*SPC*). We have found some research gaps in these algorithms which motivated us to improve them. Both, *k*-means and *S*ingle *P*ass *C*lustering (*SPC*) algorithms are implemented in *C*-language. Performance of these algorithms has been compared on artificial and real datasets using existing validity measures and indices as described by the previous researchers. It has been observed that *SPC* algorithm performs better than *k*-means algorithm over the datasets considered in this work. Unlike *k*-means algorithm, *SPC* algorithm is not sensitive to outliers; it is quite simple and does not require prior information about number of clusters and initial centroids. But, it requires threshold similarity value from the user and it is sensitive to the order of selection of data objects. To resolve the limitations of these algorithms, we have proposed a *M*odified *S*ingle *P*ass *C*lustering (*MSPC*) algorithm. It has also been implemented in *C*-language.

Threshold similarity value is not a user defined parameter in the *MSPC* algorithm. Rather, it is automatically calculated using a function of data objects. This function calculates the threshold similarity value as the mean/median of paired distance of data objects left to be clustered. Experiments have been carried out on artificial and real datasets to evaluate the performance of this algorithm. Four real datasets: Ecoli, Iris, Seeds and Wine have been

considered and quality of *k*-means, *SPC* and *MSPC* algorithms has been validated using existing validation measures and indices.

MSPC algorithm has a limitation of producing non-deterministic results since data objects are selected randomly in this algorithm. This motivated us to work on a deterministic algorithm. As such, a deterministic algorithm, *A*daptive *T*hreshold based *C*lustering (*ATC*) has also been proposed in this work. This algorithm is also implemented using *C*-language. It is based on the assumption that atleast two clusters are present in a given dataset. Using this assumption, we considered two farthest data objects as the members of two different clusters. Clusters are then generated using one of the farthest data objects. We have introduced two parameters in this algorithm to cluster the dataset. First parameter, neighborhood distance parameter is an adaptive parameter which is not specified by the user, rather, it is calculated automatically during the execution of algorithm. Second parameter, minimum support value used in this algorithm plays an important role to find significant clusters. Performance of *ATC* algorithm is assessed on artificial and real datasets: *B*reast *C*ancer *W*isconsin-*O*riginal (*BCW-O*), Ecoli, Glass Identification, Haberman's survival, Iris, Seed, Wine and Yeast. It has been observed that *ATC* algorithm is capable of detecting outliers and generates overlapped and non-overlapped clusters.

In this research work, new cluster validation measures have also been proposed for compactness and separation. Compactness measures proposed in this research work are calculated by measuring the dispersion of data objects along each dimension. For each dimension, using the mean and standard deviation of dimensions, maximum dispersed values of data objects are averaged arithmetically/geometrically. Arithmetically averaged dispersion value is named as *A*rithmetic *D*ispersion (*AD*) compactness and geometrically averaged dispersion value is named as *G*eometric *D*ispersion (*GD*) compactness. Separation measure proposed in this research work is an averaged paired distance between the data objects of clusters. It is named as *A*verage *L*inkage (*AL*) separation measure. Experimental work has been performed on artificial and real datasets to verify these measures.

Some key observations noted from the experiments conducted in this research work are:

- *MSPC* algorithm generates well separated and compact clusters as compared to *k*-means and *SPC* algorithms.

- Number of clusters generated by *MSPC* algorithm is almost same as present in the real datasets.
- *ATC* algorithm generates the clusters automatically without specifying the number of clusters beforehand.
- In *ATC* algorithm, significant clusters cover almost all the data objects of a given dataset.
- The procedure used to generate the clusters in the *ATC* algorithm is deterministic. For a given minimum support value, it gives the same members and same number of clusters on successive runs for a given dataset.
- *ATC* algorithm generates un-equal size clusters whereas *k*-means algorithm generates almost equal size clusters.
- Proposed compactness and separation measures can also be used as an alternative to existing measures.
- The proposed *M*odified *S*ingle *P*ass *C*lustering (*MSPC*) algorithm may generate illogical clusters as it selects data objects randomly.
- *MSPC* algorithm lies in the category of non-deterministic algorithms.
- The proposed algorithm, *A*daptive *T*hreshold based *C*lustering (*ATC*), is based on distance based similarity measure. So, it generates spherical clusters.
- Working of *ATC* algorithm is dependent on pair wise distance between data objects. As such, these distances are required to be computed in the beginning.

6.2 Future Scope

In future, this work can further be enriched by incorporating following concepts into the proposed algorithms:

- New methods to find a better approximation to threshold similarity value can be identified.
- Proposed algorithms (*MSPC* and *ATC*) generate spherical clusters; these can be extended to generate non-spherical clusters.

- In *ATC* algorithm while generating overlapped clusters, some new methods can be explored to decide the fuzzy membership value of overlapped objects. Therefore, this algorithm can be extended for fuzzy clustering.
- Validity measures play a pivotal role to justify the credibility of clustering algorithms. One can work on proposing even better validity measures for validation of clusters.
- Use of adjacency matrix can also be replaced by an efficient data structure.
- The proposed approach can be applied in distributed or parallel environment.

List of Publications by Author

Journal Publications:

[1] Mittal M., Sharma R. K. and Singh V. P., (2015), "Modified Single Pass Clustering with Variable Threshold Approach," International Journal of Innovative Computing, Information and Control, 11(1), 375-386.

[2] Mittal M., Sharma R. K. and Singh V. P., (2014), "Validation of *k*-means and Threshold Based Clustering Method," International Journal of Advancements in Technology, 5(2), 153-160.

[3] Mittal M., Sharma R. K. and Singh V. P., (2010), "A New Validation Method for Partitioning Based Clustering", International Journal of Computational Intelligence Research (IJCIR), RIP, ISSN 0973-1873, 6(4), 613-617.

Conference Publications:

[1] Mittal M., Sharma R. K. and Singh V. P., (2011), "Random Automatic Detection of Clusters," in Proc. IEEE Delhi Section International Conference on Image Information Processing, ICIIP-2011, JUIT Solan, 3-5th Nov. 2011.

Communicated Papers:

[1] Mittal M., Sharma R. K. and Singh V. P., "A Novel Minimum Support Based Adaptive Clustering Approach," Arabian Journal for Science and Engineering.

[2] Mittal M., Sharma R. K. and Singh V. P., "Comparison of a Threshold Based Clustering Method and *k*-means Method on Iris Dataset," Journal of Information Processing System, KIPS, Korea.

[3] Mittal M., Sharma R. K. and Singh V. P., "Survey Paper on High Dimensional Databases," International Journal of Computer Science and Network Solutions, IRAN.

References

1. Achtert E., Bohm C., Kriegel H-P, Kroger P., Muller-Gorman I. and Zimek A., (2007), "Detection and visualization of subspace cluster hierarchies," in Advances in Databases: Concepts, Systems and Applications, Springer, Berlin, Germany, pp. 152-163.
2. Aggarwal C. C., Wolf J. L., Yu P. S., Procopiuc C. and Park J. S., (1999), "Fast algorithm for projected clustering," in Proc. ACM SIGMOD International Conference on Management of Data, ACM press, pp. 61-72.
3. Aggarwal C. C. and Yu P. S., (2000), "Finding generalized projected clusters in high dimensional spaces," in Proc. ACM SIGMOD International Conference on Management of Data, ACM press, pp. 70-81.
4. Agrawal R., Gehrke J., Gunopulos D. and Raghavan P., (1998), "Automatic subspace clustering of high dimensional data for data mining applications," in Proc. ACM SIGMOD International Conference on Management of Data, Seattle, WA, pp. 94-105.
5. Agrawal R. and Srikant R., (1994), "Fast algorithms for mining association rules," in Proc. of 20th VLDB conference, pp. 487-499.
6. Ahmad A. and Dey L., (2007), "A *k*-mean clustering algorithm for mixed numeric and categorical data," Data & Knowledge Engineering, vol. 63, pp. 503-527.
7. Ankerst M., Breuing M. M., Kriegel H-P and Sander J., (1999), "OPTICS: Ordering points to identify the clustering structure," in Proc. ACM SIGMOD International Conference on Management of Data, pp. 49-60.
8. Arthur D. and Vassilvitskii S., (2007), "*k*-means++: The advantage of careful seeding," in Proc. of Symposium of Discrete Analysis, pp. 1027-1035.
9. Arya S., Mount D. M., Netanyahu N. S., Silverman R. and Wu A. Y., (1994), "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions," Journal ACM, vol. 45, no. 6, pp. 891-923.
10. Ashraf F., Ozyer T. and Alhajj R., (2008), "Employing Clustering Techniques for Automatic Information Extraction from HTML Documents," IEEE Transactions on

Systems Man Cybernetics-Part C: Applications Reviews, vol. 38, no. 5, pp. 660-673.

11. Aslam J. A., Pelekhov E. and Rus D., (1998), "Star clustering algorithm for static and dynamic information organization," Graph Algorithms and Application, vol. 8, no.1, pp. 95-129.
12. Assent I., Krieger R., Muller E. and Seidl T., (2007), "VISA: Visual subspace clustering analysis," ACM SIGKDD Explorations, vol. 9, no. 2, pp. 5-12.
13. Bagirov A. M., Ugon J. and Webb D., (2011), "Fast modified global *k*-means algorithm for incremental cluster construction," Pattern recognition, vol. 44, no. 4, pp. 866-876.
14. Balcan M. F., Blum A. and Gupta A., (2013), "Clustering under approximation stability," Journal of the ACM, vol. 60, no 2, pp. 1-34.
15. Banerjee A. and Ghosh J., (2006), "Scalable clustering algorithms with balancing constraints," Journal of Data Mining and Knowledge Discovery, vol. 13, pp. 365-395.
16. Beg M. M. S. and Ahmad N., (2007), "Web Search Enhancement by Mining User Actions," International Journal of Information Sciences, Elsevier, vol. 177, no. 23, pp. 5203-5218.
17. Bentley J. L., (1975), "Multidimensional binary search trees used for associative searching," Communications ACM, vol. 18, no. 9, pp. 509-517.
18. Berkhin P., (2006), Survey of clustering data mining techniques, Berlin: Springer, pp. 25-71.
19. Beyer K., Goldstein J., Ramakrishnan R. and Shaft U., (1998), "When is 'nearest neighbor' meaningful?," in Proc. of 7th International Conference on Database Theory (ICDT-1999), LNCS 1540, Jerusalem, Israel, pp. 217-235.
20. Boley D., Gini M., Gross R., Han E-H, Hastings K., Karypis G., Kumar V., Mobasher B. and Moore J., (2000), "Partitioning-based clustering for web document categorization," Decision Support System, vol. 27, no. 3, pp. 329-341.
21. Boutsidis C. and Magdon-Ismail M., (2013), "Deterministic feature selection for *k*-means clustering," IEEE Transactions on Information Theory, vol. 59, pp. 6099-6110.
22. Bradley P. S. and Fayyad U. M., (1998a), "Refining initial points for *k*-means clustering," Proc. of the 15th International Conference on Machine Learning

(ICML98), J. Shavlik (ed.) Morgan Kaufmann, San Francisco, pp. 91-99.

23. Bradley P. S., Fayyad U. M. and Reina C. A., (1998b), "Scaling clustering algorithms to large databases," in 4th International Conference on Knowledge Discovery and Data Mining (KDD 98), pp. 9-15.
24. Brin S., (1995), "Near Neighbor Search in Large Metric Spaces," in Proc. of the 21st International Conference on Very Large Databases (VLDB-1995), Zurich Switzerland, Morgan Kaufmann, pp. 574-584.
25. Brin S. and Page L., (1998), "The anatomy of a large-scale hypertextual web search engine," Computer networks and ISDN Systems, vol. 30, pp. 107-117.
26. Cao F., Liang J. and Jiang G., (2009), "An initialization method for the *k*-means algorithm using neighborhood model," Journal of Computers and Mathematics with Applications, vol. 58, pp. 474-483.
27. Cardot H., Cenac P. and Monnez J-M, (2012), "A fast and recursive algorithm for clustering large datasets with *k*-medians," Computational Statistics Data Analysis, vol. 56, pp. 1434-1449.
28. Celebi M. E., Kingravi H. A. and Vela P. A., (2013), "A comparative study of efficient initialization methods for the *k*-means clustering algorithm," Expert System with Applications, vol. 40, pp. 200-210.
29. Chang J-W. and Jin D-S, (2002), "A new cell-based clustering method for large, high-dimensional data in data mining applications," in Proc. of ACM symposium on Applied computing, ACM Press, pp. 503-507.
30. Chapelle O., Scholkopf B. and Zien A., (2006), Semi-supervised learning, Cambridge, MA: MIT Press.
31. Chen X., Xu X., Huang J. Z. and Ye Y., (2013), "TW-*k*-means: Automated two-level variable weighting clustering algorithm for multiview data," IEEE transactions on Knowledge and Data Engineering, vol. 25, no. 4, pp. 932-944.
32. Chen Y., Miao D. and Zhang H., (2010), "Neighborhood outlier detection," Expert systems with Applications, vol. 37, pp. 8745-8749.
33. Cheung Y-M., (2003), "*k**-means: A new generalized *k*-means clustering algorithm," Pattern Recognition Letters, Elsevier, vol. 24, no. 15, pp. 2883-2893.
34. Chiang M. M-T and Mirkin B., (2009), "Intelligent choice of the number of clusters

in *k*-means clustering: an experimental study with different cluster spreads," Journal of Classification, vol. 27, no. 1, pp. 3-40.

35. Chiu T., Fang D. P., Chen J., Wang Y. and Jeris C., (2001), "A robust and scalable clustering algorithm for mixed type attributes in large database environment," in Proc. of the International Conference on Knowledge Discovery and Data Mining (KDD01), San Francisco, CA, pp. 263-268.
36. Chu Y-H., Huang J-W, Chuang K-T, Yang D-N and Chen M-S, (2010), "Density conscious subspace clustering for high-dimensional data," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 1, pp. 16-30.
37. Cormen T. H., Leiserson C. E. and Rivest R. L. (1990), Introduction to Algorithms, Prentice Hall.
38. Damian D., Oresic M., Verheij E., Meulman J., Friedman J., Adourian A., Morel N., Smilde A. and Greef J., (2007), "Application of new subspace clustering algorithm (COSA) in medical system biology," Journal of metabolomics, vol. 3, no. 1, pp 69-77.
39. Dash M., Liu H. and Yao J., (1997), "Dimensionality reduction for unsupervised data," in Proc. of ICTAI, Newport Beach: IEEE Computer Society, pp. 532-539.
40. Davies D. L. and Bouldin D. W., (1979), "A cluster separation measure," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 1, no. 2, pp. 224-227.
41. Deelers S. and Auwatanamongkol S., (2007), "Enhancing *k*-means algorithm with initial cluster centers derived from data partitioning along the data axis with the highest variance," International Journal of Computer Science, vol. 2, no. 4, pp. 323-328.
42. Dhillon I. S. and Modha D. S., (2001), "Concept Decompositions for Large Sparse Text Data using Clustering," Machine Learning, vol. 42, no. 1, pp. 143-175.
43. Donoho D. L., (2000), "High Dimensional Data Analysis: The curses and blessings of dimensionality," American Math. Society Conference: Mathematical Challenges of the 21st Century, Los Angeles, CA, August, pp. 6-11.
44. Dubes R. and Jain A. K., (1976), "Clustering techniques: The user's dilemma," Pattern Recognition, vol. 8, no. 4, pp. 247-260.

45. Duda R. O. and Hart P. E., (1973), Pattern classification and scene analysis, John Wiley and Sons, New York.
46. Dunn J. C. (1974), "Well Separated Clusters and Optimal Fuzzy Partitions," Journal of Cybernetics, vol.4, pp. 95-104.
47. El-Hamdouchi A. and Willett P., (1989), "Comparison of Hierarchic Agglomerative Clustering Methods for Document Retrieval," The Computer Journal, vol. 32, no. 3, pp. 220-227.
48. Erisoglu M., Calis N. and Sakallioglu S., (2011), "A new algorithm for initial cluster centers in *k*-means algorithm," Pattern Recognition Letters, vol. 32, no. 14, pp. 1701-1705.
49. Ertöz L., Steinbach M. and Kumar V., (2001), "Finding topics in collections of documents: A shared nearest neighbor approach," Clustering and Information Retrieval, vol. 11, pp. 83-103.
50. Ester M., Kriegel H-P, Sander J. and Xu X., (1996), "A Density based algorithm for discovering clusters in large spatial databases with noise," in Proc. of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, AAAI Press, Portland, pp. 226-231.
51. Fahim A. M., Saake G., Salem A. M., Torkey F. A. and Ramadan M. A., (2008a), "*k*-means for Spherical Clusters with Large Variance in Sizes," in Proc. World Academy Science, Engineering and Technology, vol. 35, pp. 177-182.
52. Fahim A. M., Saake G., Salem A. M., Torkey F. A. and Ramadan M. A., (2008b), "DCBOR: A density clustering based on outlier removal," in Proc. World Academy Science, Engineering and Technology, vol. 35, pp. 171-176.
53. Fahim A. M., Salem A. M., Torkey F. A. and Ramadan M. A., (2006), "An efficient enhanced *k*-means clustering algorithm," Journal of Zhejiang University Science A, vol. 7, no. 10, pp 1626-1633.
54. Fayyad U. M., Piatetsky-Shapiro G., Smyth P., (1996), "From Data mining to Knowledge Discovery in Databases," AAAI/MIT Press, pp. 37-54.
55. Freund Y. and Schapire R. E., (1997), "A decision-theoretic generalization of on-line learning and an application to boosting," Journal of computer system science, vol. 55, no. 1, pp. 119-139.

56. Friedman J. H., Bentley J. L. and Finkel R. A., (1977), "An algorithm for finding best matches in logarithmic expected time," ACM transactions on Mathematical Software, vol. 3, no. 3, pp. 209-226.
57. Friedman J. H. and Meulman J. J., (2002), "Clustering objects on subsets of attributes," Journal of the royal statistical society: series B (statistical methodology), vol. 66, no.4, pp. 815-849.
58. Gil-Garcia R. J., Badia-Contelles J. M. and Pons-Porrata A., (2003), "Extended Star Clustering," LNCS 2905, Springer, pp. 480-487.
59. Goil S., Nagesh H. and Choudhary A., (1999), "MAFIA: Efficient and scalable subspace clustering for very large data sets," Northwestern university, 2145 Sheridan road, Evanston IL 60208, Tech. Rep. CPDC-TR-9906-010.
60. Guha S., Rastogi R. and Shim K., (1999), "Rock: A robust clustering algorithm for categorical attributes," International conference on data engineering, Sydney, Australia, March 1999, pp. 512-521.
61. Guha S., Rastogi R. and Shim K., (1998), "CURE: An efficient clustering algorithm for large databases," in Proc. ACM SIGMOD International Conference on Management of Data, pp. 73-84.
62. Halkidi M., Vazirgiannis M. and Batistakis I., (2000), "Quality scheme assessment in the clustering process," Principles Data Mining Knowledge Discovery, LNCS, vol. 1910, pp. 265-276.
63. Halkidi M., Batistakis Y. and Vazirgiannis M., (2001), "On clustering validation techniques," Journal of Intelligent Information Systems, vol. 17, no. 2/3, pp. 107-145.
64. Halkidi M., Batistakis Y. and Vazirgiannis M., (2002a), "Cluster validity methods: Part I," SIGMOD Rec., vol. 31, no. 2, pp. 40-45.
65. Halkidi M., Batistakis Y. and Vazirgiannis M., (2002b), "Cluster validity checking methods: Part II," SIGMOD Rec., vol. 31, pp. 19-27.
66. Han J., Cheng H., Xin D. and Yan X., (2007), "Frequent pattern mining: current status and future directions," Data Mining Knowledge Discovery (DATAMINE), vol. 15, no. 1, pp.55-86.
67. Han E-H., Karypis G., Kumar V. and Mobasher B., (1997), "Clustering in a high-dimensional space using hypergraph models," Department of Computer Science,

University of Minnesota, Minneapolis, Minnesota, Tech. Rep. TR-97-063.

68. Han J. and Kamber M., (2006), Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers.
69. Harper A., Jin Z., Sokunle R. and Wadhwa M., (2013), "Price volatility in the silver spot market: an empirical study using Garch applications," Journal of Finance and Accountancy, vol. 13, pp. 1-11.
70. Hasan M. A., Chaoji V., Salem S. and Zaki M. J., (2009), "Robust partitional clustering by outlier and density insensitive seeding," Pattern Recognition Letters, vol. 30, pp. 994-1002.
71. Hasan M. A., Salem S., Pupacdi B. and Zaki M. J., (2009), "Clustering with a lower bound on similarity," PAKDD 2009, LNAI 5476, pp. 122-133.
72. Hastie T. and Tibshirani R., (1996), "Discriminant adaptive nearest neighbor classification," IEEE transaction on pattern analysis machine intelligence, vol. 18, no. 6, pp. 607-616.
73. Hatamlou A., Abdullah S. and Nezamabadi-pour H., (2012), "A combined approach for clustering based on *k*-means and gravitational search algorithms," Swarm and Evolutionary Computations, vol. 6, pp. 47-52.
74. Hautamaki V., Cherednichenko S., Karkkainen I., Kinnunen T. and Franti P., (2005), "Improving *k*-means by outlier removal," SCIA, LNCS 3540, pp. 978-987.
75. Hinneburg A. and Keim D., (1998), "An efficient approach to clustering in large multimedia databases with noise," in Proc. 4th International Conference on Knowledge Discovery and Data Mining, New York, pp. 58-65.
76. Hinneburg A. and Keim D., (1999), "Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering," in Proc. 25th International Conference on Very Large Data Bases (VLDB-1999), Edinburgh, Scotland, Morgan Kaufmann, pp. 506-517.
77. Hodge V. J. and Austin J., (2004), "A survey of outlier detection methodologies," Artificial intelligence review, vol. 22, no. 2, pp. 85-126.
78. Hoppner F., Klawonn F., Kruse R. and Runkler T., (1999), Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition, John Wiley & Sons.

79. http://archive.ics.uci.edu/ml/datasets.html
80. Huang Z., (1998), "Extensions to the *k*-means algorithm for clustering large data sets with categorical values," Data Mining Knowledge Discovery, vol. 2, pp. 283-304.
81. Jain A. K., (2010), "Data clustering: 50 years beyond *k*-means," Pattern Recognition Letters, vol. 31, no. 8, pp. 651-666.
82. Jain A. K., Murty M. N. and Flynn P. J., (1999), "Data clustering: a review," ACM Computing Surveys (CSUR), vol. 31. no. 3, pp. 264-323.

    Jiang F., Sui Y. and Cao C., (2009), Some issues about outlier detection in rough set theory," Expert system with Applications, vol. 36, pp. 4680-4687.
83. Jiang F., Sui Y. and Cao C., (2009), Some issues about outlier detection in rough set theory," Expert system with Applications, vol. 36, pp. 264-323.
84. Kailing K., Kriegel H-P and Kroger P., (2004), "Density-connected subspace clustering for high-dimensional data," in Proc. 4th SIAM International Conference on Data Mining, Orlando, FL, pp. 246-257.
85. Kanungo T., Mount D. M., Netanyahu N. S., Piatko C. D., Silverman R. and Wu A. Y., (2002), "An Efficient *k*-means clustering algorithm: Analysis and implementation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 881-892.
86. Karypis G. and Han E-H., (2000), "Concept Indexing: A fast dimensionality reduction algorithm with applications to document retrieval & categorization," 9th International Conference on Information and Knowledge Management (CIKM 2000), McLean, VA.`
87. Karypis G., Han E-H and Kumar V., (1999), "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling," IEEE Computer, vol. 32, no. 8, pp. 68-75.
88. Karypis G. and Kumar V., (1998), "hMETIS 1.5: A hypergraph partitioning package," Department of Computer Science, University of Minnesota, Tech. Rep.
89. Katsavounidis I., Kuo C-C. J. and Zhang Z., (1994), "A new initialization technique for generalized Lloyd iteration," IEEE Signal Processing Letters, vol. 1, no. 10, pp. 144-146.
90. Kaufman L. and Rousseeuw P. J., (1990), Finding groups in data: An introduction to

cluster analysis, John Wiley & Sons.

91. Kaufman L. and Rousseeuw P. J., (1987), "Clustering by means of medoids," in Statistical Data Analysis based on the L_1 norm, Amsterdam, pp. 405-416.
92. Khan S. S., and Ahmad A., (2004), "Cluster center initialization algorithm for *k*-means clustering," Pattern Recognition Letter, vol. 25, no. 11 , pp. 1293-1302.
93. Kim S., Cho N. W., Kang B. and Kang S-H, (2011), "Fast outlier detection for very large log data," Expert Systems with Applications, vol. 38, pp. 9587-9596.
94. Lai J. Z. C., Huang T-J, and Liaw Y-C, (2009), "A fast *k*-means clustering algorithm using cluster center displacement," Pattern Recognition, vol. 42, pp. 2551-2556.
95. Lai J. Z. C. and Huang T-J, (2010), "Fast global *k*-means clustering using cluster membership and inequality," Pattern Recognition, vol. 43, pp. 1954-1963.
96. Lai J. Z. C., and Liaw Y-C, (2008), "Improvement of the *k*-means clustering filtration algorithm," Pattern Recognition, vol. 41, pp. 3677-3681.
97. Li C. S., (2011), "Cluster center initialization method for *k*-means algorithm over data sets with two clusters," in Proc. International Conference on Advances in Engineering, Elsevier, vol. 24, pp. 324-328.
98. Likas A., Vlassis N. and Verbeek J. J., (2003), "The global *k*-means clustering algorithm," Pattern Recognition Letters, vol. 36, pp. 451–461.
99. Liu B., Xia Y. and Yu P. S., (2000), "Clustering through decision tree construction," IBM Research Report RC 21695 (97737), vol. 20 March 2000, pp. 1-20.
100. Lloyd S. P., (1982), "Least squares quantization in PCM," IEEE Transactions on Information Theory, vol. 28, pp. 129–137.
101. MacQueen J., (1967), "Some methods for classification and analysis of multivariate observations," in Proc. 5th Symposium Mathematical Statistics and Probability, Berkeley, CA, vol. 1, pp. 281-297.
102. Mavroeidis D. and Marchiori E., (2014), "Feature selection for *k*-means clustering stability: theoretical analysis and an algorithm," Data Mining and Knowledge Discovery, vol. 28, no. 4, pp. 918-960.
103. McCullagh P. and Yang J., (2008), "How many clusters?," Bayesian Analysis, vol. 3, no. 1, pp. 101–120.
104. McLachlan G. J. and Krishnan T., (1997), The EM algorithm and extensions, Wiley,

New York.

105. Melnykov I. and Melnykov V., (2014), "On *k*-means algorithm with the use of Mahalanobis distances," Statistics and Probability Letters, vol. 84, pp.88-95.
106. Merz P., (2003), "An Iterated Local Search Approach for Minimum Sum of Squares Clustering," IDA, pp. 286-296.
107. Milenova B. L. and Campos M. M., (2002), "O-Cluster: scalable clustering of large high dimensional data sets," in Oracle Data Mining Technologies, Oracle Corporation, pp. 290-297.
108. Minz, S. and Jain R., (2005), "Refining decision tree classifiers using rough set tools," International Journal of Hybrid Intelligent Systems, vol. 2, pp. 133-148.
109. Mitra P., Murthy C. A. and Pal S. K., (2002), "Unsupervised feature selection using feature similarity," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 3, pp. 301-312.
110. Murtagh F., Starck J-L, and Berry M. W., (2000), "Overcoming the curse of dimensionality in clustering by means of the wavelet transform," The Computer Journal, vol. 43, pp. 107-120.
111. Ng R. T. and Han J., (1994), "Efficient and effective clustering methods for spatial data mining," in Proc. 20th international conference on very large databases, Santiago, Chile, Morgan Kaufmann Publishers, San Francisco, California, pp. 144-155.
112. Ng R. T., and Han J., (2002), "CLARANS: A method for clustering objects for spatial data mining," IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 5, pp. 1003-1016.
113. Nguyen D. T., Chen L. and Chan C. K., (2012), "Clustering with Multiviewpoint-Based similarity Measure," IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 6, pp. 988-1001.
114. Panigrahi S. K., Chakraverty S. and Mishra B. K., (2009), "Vibration based damage detection in a uniform strength beam using genetic algorithm," Meccanica, vol. 44, pp. 697-710.
115. Park H-S. and Jun C-H, (2009), "A simple and fast algorithm for *k*-medoids clustering," Expert Systems with Applications, vol. 36, no. 2, pp. 3336-3341.
116. Parsons L., Haque E. and Liu H., (2004), "Subspace clustering for high dimensional

data: a review," ACM SIGKDD Explor Newslett, vol. 6, no. 1, pp. 90-105.

117. Patrikainen A. and Meila M., (2006), "Comparing subspace clusterings," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 7, pp. 902-916.
118. Pelleg D. and Moore A., (2000), "X-means: Extending *k*-means with efficient estimation of the number of clusters," in Proc. ICML-2000, pp. 727-734.
119. Pelleg D. and Moore A., (1999), "Accelerating exact *k*-means algorithms with geometric reasoning," in Proc. 5th International Conference on Knowledge Discovery in Databases, AAAI Press, pp. 277-281.
120. Pham D. T., Dimov S. S. and Nguyen C. D., (2004), "Selection of *k* in *k*-means clustering," Journal of Mechanical Engineering Science, 219, pp. 103-119.
121. Pilevar A. H. and Sukumar M., (2005), "GCHL: A grid-clustering algorithm for high-dimensional very large spatial data bases," Pattern Recognition Letters, vol. 26, no. 7, pp. 999-1010.
122. Procopiuc C. M., Agarwal P. K., Jones M. and Murli T. M., (2002), "A montecarlo algorithm for fast projective clustering," in Proc. ACM SIGMOD International Conference on Management of Data, ACM press, pp. 418-427.
123. Quinlan J. R., (1993), C4.5: Programs for machine learning. Morgan Kaufmann Publishers, San Mateo.
124. Rahman A. and Verma B., (2013), "Ensemble classifier generation using non-uniform layered clustering and Genetic Algorithm," Knowledge-Based Systems, vol. 43, pp. 30-42.
125. Ray S. and Turi R., (1999), "Determination of number of clusters in *k*-means clustering and application in colour image segmentation," in Proc. 4th International Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99), India, pp. 137-143.
126. Reddy D. and Jana P. K., (2012), "Initialization for *k*-means clustering using voronoi diagram," Procedia Technology, vol. 4, pp. 395-400.
127. Redmond S. J. and Heneghan C., (2007), "A method for initialising the *k*-means clustering algorithm using *k-d* trees," Pattern Recognition Letters, vol. 28, no. 8, pp. 965-973.
128. Rijsbergen C. J. V., (1979), A new theoretical framework for information retrieval,

Butterworth, London, 2nd ed.

129. Rose K., (1998), "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," in Proc. IEEE, vol. 86, no. 11, pp. 2210-2239.
130. Rousseeuw P. J., (1987), "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," Journal of Computational and Applied Mathematics, vol. 20, no. 1, pp. 53-65.
131. Saabni R., Asi A. and El-Sana J., (2014), "Text line extraction for historical document images," Pattern Recognition Letters, vol. 35, pp. 23-33.
132. Salton G. and Wong A., (1978), "Generation and search of clustered files," ACM TODS, vol 3, pp. 321-346.
133. Salton G., (1971), The SMART Retrieval System, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
134. Savaresi S. M. and Boley D. L., (2001), "On the Performance of Bisecting *k*-means and PDDP," in Proc. 1st International SIAM Data Mining Conference, Chicago, IL.
135. Scitovski R. and Sabo K., (2014), "Analysis of the *k*-means algorithm in the case of data points occurring on the border of two or more clusters," Knowledge-Based Systems, vol. 57, pp. 1-7.
136. Steinbach M., Karypis G. and Kumar V., (2000), "A comparison of document clustering techniques," Department Comp. Sci. Engg., University of Minnesota, Tech. Rep.00-034, Minneapolis, USA.
137. Tibshirani R., Walther G. and Hastie T., (2001), "Estimating the number of clusters in a data set via the gap statistic," Journal Royal Statistical Society: Series B, vol. 63, no. 2, pp. 411-423.
138. Toshniwal D. and Joshi R. C., (2005), "Using Cumulative Weighted Slopes for Clustering Time Series Data," GESTS International Transaction on Computer Science and Engineering, vol 20, no. 1, pp. 29-40.
139. Vapnik V. N., (1999), "An overview of statistical learning theory," IEEE transaction on neural networks, vol. 10, no. 5, pp. 988-999.
140. Wang L., Leckie C., Ramamohanarao K. and Bezdek J., (2009), "Automatically Determining the Number of Clusters in Unlabeled Data Sets," IEEE transactions on

Knowledge and Data Engineering, vol. 21, no. 3, pp. 335-350.

141. Wang W., Yang J. and Muntz R., (1997), "STING: A statistical information grid approach to spatial data mining," in Proc. 23rd VLDB conference, Athens, Greece.
142. Woo K. G. and Lee J-H, Kim M-H. and Lee Y-J, (2002), "FINDIT: Afast and intelligent subspace clustering algorithm using dimension voting," Information and Software Technology," vol. 46, no. 4, pp. 255-271.
143. Wu X., Kumar V., Quinlan J-R, Ghosh J., Yang Q., Motoda H., McLachlan G. J., Ng A., Liu B., Yu P. S., Zhou Z-H, Steinbach M., Hand D. J. and Steinberg D., (2008), "Top 10 algorithm in data mining," Knowledge and Information Systems, vol. 14, pp. 1-37.
144. Xiong H., Wu J. and Chen J., (2009), "*k*-means clustering versus validation measures: A data-distribution perspective," IEEE transactions on systems, man and cybernetics-Part B, vol. 39, no. 2, pp. 318-331.
145. Yan M., (2005), "Methods of Determining the Number of Clusters in a Data Set and a New Clustering Criterion," Ph.D. Dissertation, Faculty Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
146. Yang J., Wang W., Wang H. and Yu P., (2002), "δ-clusters: Capturing subspace correlation in a large data set," in Proc. 18th International Conference on Data Engineering, pp. 517-528.
147. Yang P. and Zhu Q., (2010), "Finding key attribute subset in dataset for outlier detection," Knowledge-Based Systems, vol. 24, no. 2, pp. 269-274.
148. Yuan F., Meng Z-H, Zhang H-X and Dong C-R, (2004), "A new algorithm to get the initial centroids," in Proc. International Conference on Machine Learning and Cybernetics, vol.2, pp.1191-1193.
149. Zaki M. J., Peters M., Assent I. and Seidl T., (2007), "CLICKS: An effective algorithm for mining subspace clusters in categorical datasets," Data and Knowledge Engineering, vol. 60, no. 1, pp. 51-70.
150. Zalik K. R., (2008), "An efficient *k*-means clustering algorithm," Pattern Recognition Letters, vol. 29, pp. 1385-1391.
151. Zhang T., Ramakrishnan R. and Livny M., (1996), "BIRCH: An efficient data clustering method for very large databases," in Proc. ACM SIGMOD Conference on

Management of Data, Montreal, Canada: ACM, New York, pp. 103-114.

152. Zhang Y. J. and Cheng E., (2013), "An Optimized Method for Selection of the Initial Centers of *k*-means Clustering," Integrated Uncertainty in Knowledge Modeling and Decision Making, LNCS, vol. 8032, pp. 149-156.