T 143

50505

T 143

27/02/13

50505

26/12/11

Intrinsic Pattern Identification Using Clustering Technique

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Sauravjyoti Sarmah
Registration No 011 of 2010

School of Engineering
Department of Computer Science and Engineering
Tezpur University
2010

Abstract

Cluster analysis has been widely used over various domains to identify similar groups or patterns inherent in the data. The aim of this thesis is to study different clustering techniques applicable in spatial and gene expression data. For spatial data domain, this thesis presents three clustering techniques. The first technique (GDCT) is a grid-density based technique for identifying clusters of arbitrary shapes from 2D spatial data even in presence of noise. We succeeded in detecting the various types of outliers by using a outlier detection technique in-built with GDCT. The next two techniques (SATCLUS and GDSDC) are used to detect clusters in satellite data in a two phase process. Both SATCLUS and GDSDC use a grid-density based technique to identify the coarse clusters in the first phase. In the second phase, SATCLUS uses a partitioning strategy and GDSDC uses a fuzzy approach to obtain the final clusters from the coarse clusters. Also, to handle massive datasets, we propose two distributed techniques for 2D spatial data and satellite imagery. A study of clustering techniques for the analysis of gene expression data has also been discussed in this thesis. This thesis also includes two gene expression data clustering techniques. The first one is a density based clustering technique (GenClus) which clusters genes without taking the number of clusters as an input parameter. This thesis also incorporates an incremental version of the GenClus (InGenClus) to handle incremental gene expression data. The second technique GeneClusTree uses a hierarchical and density based approach to cluster the genes and forms a tree structure which helps in the visualization of the results. The proposed algorithm has been validated on several real-life datasets and found to perform well in comparison to similar algorithms. All clustering algorithms have been validated using various statistical measures.

Keywords — Clustering, proximity measure, density based clustering, grid based clustering, embedded clusters, satellite data, coherent pattern, co-expressed gene, hierarchical clustering, distributed clustering

Tezpur University

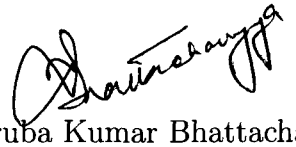
Certificate

This is to certify that the thesis entitled "Intrinsic Pattern Identification Using Clustering Technique" submitted to the Tezpur University in the Department of Computer Science and Engineering under the School of Engineering in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science is a record of research work carried out by Mr. Sauravjyoti Sarmah under my personal supervision and guidance.

All helps received by him from various sources have been duly acknowledged.

No part of this thesis has been reproduced elsewhere for award of any other degree.

Signature of Research Supervisor



(Dhruba Kumar Bhattacharyya)
Designation: Professor
School: Engineering
Department: Computer Science and Engineering

Declaration

I, Sauravjyoti Sarmah, hereby declare that the thesis entitled "*Intrinsic Pattern Identification Using Clustering Technique*" submitted to the Department of Computer Science and Engineering under the School of Engineering, Tezpur University, in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy, is based on bona fide work carried out by me. The results embodied in this thesis have not been submitted in part or in full, to any other university or institute for award of any degree or diploma.

Sauravjyoti Sarmah.

(Sauravjyoti Sarmah)

Acknowledgements

I want to express my deep gratitude to my thesis supervisor, Professor D.K. Bhattacharyya for his devoted guidance, advices, support and endless patience throughout the course of this research. With his excellent research experience, he steered me to the research on intrinsic pattern identification using clustering techniques. This thesis would not have been finished without his supervision and meticulous attention to details.

I owe a debt of gratitude to my wife for her suggestions and fruitful discussions.

I am highly grateful to Prof. J. K. Kalita for his constructive comments and insightful suggestions. I am also very thankful to the rest of my thesis guidance committee, including Prof. M. Dutta and Dr. S.M. Hazarika. Their advice and suggestions have been very helpful.

I would also like to thank NESAC, Umium, Meghalaya, for providing me some of the satellite images that have been used in this thesis. The work on satellite images is an outcome of a research project funded by ISRO under RESPOND scheme.

I am grateful to all the technical and non-technical members of the Department for their support.

My deep gratitude goes to my family members, colleagues and my friends. I can never express my thanks enough for their endless love, support and understanding.

Tezpur University

Certificate

This is to certify that the thesis entitled "Intrinsic Pattern Identification Using Clustering Technique" submitted by Mr. Sauravjyoti Sarmah to Tezpur University in the Department of Computer Science and Engineering under the School of Engineering in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science has been examined by us on _____ and found to be satisfactory.

The Committee recommends for award of the degree of Doctor of Philosophy.

Signature of

Principal Supervisor

External Examiner

Date:

Contents

List of Tables

List of Figures

Notations Used in this Thesis

$d_{x,y}$	:	Dissimilarity between object x and object y.
$S_{x,y}$	:	Similarity between objects x and y.
D	:	Database.
N	:	Total number of objects in dataset D.
k	:	Total number of clusters.
$min\_d_{x,y}$	:	Minimum dissimilarity between object x and object y.
$max\_d_{x,y}$	:	Maximum dissimilarity between object x and object y.
$min\_S_{x,y}$	:	Minimum similarity between objects x and y.
$max\_S_{x,y}$	:	Maximum similarity between object x and object y.
q_{10}	:	Number of variables with vfile:///usr/share/doc/HTML/index.htmlalue 1 for the i^{th} object and 0 for the j^{th} object
q_{01}	:	Number of variables with value 0 for the i^{th} object and 1 for the j^{th} object
q_{00}	:	Number of variables with value 0 for the i^{th} object and 0 for the j^{th} object
q_{11}	:	Number of variables with value 1 for the i^{th} object and 1 for the j^{th} object
$\rho_{i,j}$	:	Pearson's correlation coefficient between objects i and j.
SMC	:	Simple Matching Coefficient.
J	:	Jaccard coefficient.
n	:	Number of dimensions.
$Jacknife_{i,j}$	:	Jacknife correlation coefficient between objects i and j.
k_p	:	Total number of processors.
$gr_n \times gr_n$	:	Total number of grid cells.
ε	:	Radius.
$MinPts$	:	Minimum number of objects.
ε'	:	Core distance.
l^i	:	Number of iterations.
Sz	:	Size of sample.
m_m	:	Maximum number of neighbors.
m_a	:	Average number of neighbors.

β'	:	Density confidence between two cells in GDCT.
$d_n(p_1)$	:	Density of cell p_1.
$d_n(T_{p_1})$	:	Density of triangle T_{p_1}.
C_i	:	The i^{th} cluster.
no_p	:	The set of objects in cell p.
p_{ij}	:	The j^{th} neighbor of cell p_i, where $j = 1, \cdots, 8$.
$T_{p_{iv}}$	:	v^{th} triangle inside the cell p_i, where $v = 1, \cdots, 4$.
L	:	Line outlier.
S	:	Single linkage outlier.
ξ'	:	Minimum threshold for group outlier.
γ	:	Minimum number of cells in line outlier.
η	:	Parameter for line outlier, $\eta = 2$.
ρ'	:	Minimum number of cells in single linkage outlier, $\rho' = 5$.
δ	:	Threshold for density variance of a core object.
τ	:	Threshold for homogeneity of density variation.
α_D	:	Density threshold of DDSC algorithm.
CDV_α	:	Threshold for cluster file:///usr/share/doc/HTML/index.htmldensity variance.
CSI_λ	:	Threshold for cluster similarity index.
$LOFUB$	:	Minimum number of points for an object to be core in LDBSCAN algorithm.
pct	:	Threshold for controlling the fluctuation of local-density.
$Minpts_{LOF}$	:	Minimum number of points for calculating LOF.
$Minpts_{LDBSCAN}$	:	Minimum number of points required for the clustering algorithm LDBSCAN.
$P_o(p)$	:	Population-object ratio of cell p.
ω	:	Confidence between two cells in SATCLUS and GDSDC.
$cell_p$	:	Total number of cells in a cluster.
u_{ij}	:	Membership degree of x_j to cluster C_i.
m_f	:	Fuzziness parameter.

H_{avg}	:	Average homogeneity.
β	:	Cluster validity index.
D^*	:	Grid mesh.
D_i	:	i^{th} Partition.
P_i	:	i^{th} Processor.
C^l	:	Number of clusters detected in local nodes.
D_G	:	Gene database.
G^*	:	Set of all genes in D_G.
T^*	:	Set of all Conditions in D_G.
G	:	Total number of Genes.
T	:	Total number of Conditions.
ε_{g_i,t_j}	:	Expression value of gene g_i at condition t_j.
$\xi_{i,j}$	:	Discretized value of gene g_i at condition t_j.
$\wp_{g_i}$	:	Regulation pattern of g_i.
Max_{EV}	:	Maximum expression value.
Min_{EV}	:	Minimum expression value.
N_{level}	:	Neighborhood level of gene g_i.
$level$	:	Dynamically calculated parameter.
$range\_id(g_i, t_k)$	:	Range value of gene g_i at condition t_k.
$range\_id(g_i)$	:	Pattern of range values of gene g_i.
S_i	:	i^{th} sub-cluster.
$\alpha_{\varepsilon_{g_i,t_j},\varepsilon_{g_i,t_{j+1}}}$	:	Angle information of gene g_i for condition t_j.
α'_{g_j}	:	Angle_id pattern of gene g_j.
$\hbar$	:	Set of conditions over which genes match.
θ^*	:	Height of the tree.
σ	:	Minimum number of genes in the neighborhood of an initiator.
δ'	:	Tuning parameter.
n_i	:	i^{th} node of GeneClusTree.
RV_{ni}	:	Reference vector of the i^{th} node.

Chapter 1

Introduction

Data mining also known as knowledge discovery is the process of analyzing data from different perspectives and summarizing it into useful information. Data mining is the extraction of hidden previously unknown and potentially useful information from large databases. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Data mining is defined in [HMS04] as follows:

> *Data mining is the analysis of (often large) observational datasets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.*

Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. Data mining tools can answer business questions that traditionally are too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations or is too difficult to find.

1.1 Data Mining Tasks

Data mining tasks are categorized into different types based on the types of models or patterns they find. Generally data mining tasks are of two types: predictive and descriptive. Predictive mining tasks perform inference on the current data in order

to make predictions. Descriptive mining tasks characterize general properties of the data in the database. Some important data mining tasks according to [HK06] are given below.

1. *Mining frequent patterns, Association and Correlations*: Mining patterns that occur frequently in data leads to the discovery of interesting associations and correlations within data.

2. *Classification and prediction*: Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts. The purpose is to use the model to predict the class of an object whose class label is unknown. The derived model is based on the analysis of training data (i.e., data objects whose class labels are unknown).

3. *Outlier analysis*: A database may contain data objects whose characteristics are significantly different from the rest of the data. These data objects are known as outliers. In some applications such as fraud detection, rare events can be more interesting than regular ones. The analysis of outlier data is termed *outlier mining.*

4. *Evolution analysis*: Data evolution analysis describes and models trends or regularities for objects whose behavior changes over time.

5. *Cluster analysis*: Cluster analysis groups data objects based on information found in the data that describes the objects and their relationships. The goal is to partition a set of objects into groups, so that objects with similar characteristics are grouped together and different groups contain objects with dissimilar characteristics. The greater the similarity within a group and the greater the difference between groups, the better or more distinct is the clustering.

Our work focuses on clustering and therefore we concentrate on cluster analysis only.

1.2 Clustering and its Importance

Cluster analysis is an important task in data mining. It is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are

similar in some sense. The key idea is to identify classications of the objects that is useful for the specific aims of the analysis. This idea has been applied in many areas including astronomy, remote sensing, biology, archeology, medicine, chemistry, education, psychology, linguistics and sociology. Clustering is an unsupervised learning method, and a common technique for statistical data analysis used in many fields including machine learning, data mining, pattern recognition, image analysis and bioinformatics.

In recent years, clustering methods have been used extensively in analyzing spatial data, satellite imagery and biological data, especially from DNA microarrays measurements. The purpose of this work is to study clustering of data with numeric attribute values. For our study, we use three different data domains namely spatial 2D data, satellite imagery and gene expression data.

1.2.1 Types of Data

Datasets may differ in a number of ways. For example, the attributes for describing data objects can be of different types and datasets may have special characteristics e.g., time series data which have explicit relationship between one data and another. The different types of attributes for describing data objects [HK06] are as follows:

1. Interval-Scaled Variables: These are continuous measurements of a roughly linear scale. Here differences between values are meaningful. Example: weight, height, latitude, longitude, temperature etc.

2. Nominal Variable: These are just different names; provides enough information to distinguish one from the other. Examples include ID numbers, eye color, zip codes, gender.

3. Binary Variables: These variables can take either of two states: 0 or 1, where 0 means the variable is absent and 1 means the variable is present.

4. Categorical Variables: These variables are a generalization of the binary variables where it can take more than two states. For example, map color may take five states: red, yellow, green, blue and pink.

5. Ordinal Variables: These variables resemble categorical variables except that the states must be ordered in a meaningful sequence. For example, the medals won in a sporting event <gold, silver, bronze>, rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in <tall, medium, short>.

6. Ratio-Scaled Variables: These variables make positive measurements on a non-linear scale such as an exponential scale. Here both differences and ratios are meaningful. Examples include temperature in Kelvin, length, time, counts.

7. Mixed-type Variables: These variables may be mixture of the various types of variables mentioned above.

General Characteristics of Datasets

The main characteristics that apply to datasets and have a significant impact on the data mining techniques used are [TSK09] given below.

1. Dimensionality: The dimensionality of a dataset is the number of attributes that the objects in the dataset possess. Analyzing high dimensional data is difficult due to the curse of dimensionality [HK06].

2. Sparsity: Only presence of data in the dataset counts. For example, in a numeric dataset, most of the attributes of an object may consist of 0 values i.e., fewer than 1% of the entries in the dataset may be non-zero. Therefore, non-zero values only need to be stored and processed reducing the computation time and storage space required.

3. Resolution: Properties of data obtained at different levels of resolution are different i.e., the patterns in the data are also dependent on the level of resolution. A pattern may not be visible at a finer level of resolution but at coarser resolution it might be visible. However, if the resolution is too coarse the pattern may disappear.

In data mining, data are available from different sources and applications. A few of them are mentioned below.

1. Record Data: Data that consists of a collection of records, each of which consists of a fixed set of attributes

    - Data Matrix: If data objects have the same fixed set of numeric attributes, the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute. Such dataset can be represented by an $N \times T$ matrix, where there are N rows, one for each object, and T columns, one for each attribute.
    - Document Data: Each document is represented as a 'term' vector where each term is a component (attribute) of the vector. The value of each component is the number of times the corresponding term occurs in the document.
    - Transaction Data: A special type of record data, where each record (transaction) involves a set of items. For example, consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that are purchased are the items.

2. Graph Data: A graph data structure consists mainly of a finite (and possibly mutable) set of pairs, called edges or arcs, of certain entities called nodes or vertices. Much data mining research is focused on algorithms that can discover concepts in non-relational data represented using only an entity's attributes. However, much of the data collected is relational, or structural, in nature, requiring tools for the analysis and discovery of concepts in structural data. Graphs provide a natural representation for many of these structured data applications. Graph mining is becoming increasingly popular in recent years because of numerous applications some of which are presented below. A detailed discussion on various kinds of graph mining algorithms is found in [CH07].

    - Chemical Data: Chemical data is often represented as graphs in which the nodes correspond to atoms, and the links correspond to bonds between the atoms. In some cases, substructures of the data may also be used as individual nodes [AW10].

- Biological Data: Biological data is modeled in a similar way as chemical data. However, the individual graphs are typically much larger. Furthermore, the nodes are typically represent specific entities of the biological models. A typical example of a node in a DNA application is an amino-acid. A single biological network may easily contain thousands of nodes [AW10].
- Networked and Web Data: In the case of computer networks and the web, the number of nodes in the underlying graph may be massive. Since the number of nodes is massive, it may lead to a very large number of distinct edges. This is also referred to as the massive domain issue in networked data [AW10].
- XML data: XML data is a natural and general representation of graph data. We note that mining and management algorithms for XML data are also quite useful for graphs, since XML data can be viewed as labeled graphs. In addition, the attribute-value combinations associated with the nodes make the problem much more challenging [AW10].

3. Ordered Data: Ordered data indicate that the columns contain values that define a sequence or order. It contains a sequence of transactions. Different forms of ordered data include the following.

   - Spatial Data: Spatial data may be viewed as consisting of objects with some location in a physical space.
   - Temporal Data: Data stored in a temporal database have a time period attached to it. A temporal database contains built-in time infrastructure, e.g. a temporal data model and a temporal version of Structured Query Language. More specifically the temporal infrastructure usually includes stamping with transaction-time.
   - Sequential Data: A sequence is an ordered set of elements and often arise through measurement of time series phenomena. Each element can be numerical, categorical or of mixed type. The length of a sequence is not fixed. The order is determined by time or position and can be regular or irregular. Applications of this type of data include speech (sequence

of phonemes), language (sequence of words and delimiters), handwriting (sequence of strokes), bioinformatics (A genes sequence has 4 possible nucleotides, example: AACTGACCTGGGCCCAATCC; A protein sequence has 20 possible amino acids, example: MAQQWSLQRLAGRHPQDSYEDSTQSSIFIYTNSNSTRGPFEGPNYHIAPR), telecommunications (alarms, data packets), retail data mining (customer behavior, sessions in an e-store (example, Amazon)) and intrusion detection.

To form groups of similar objects, a measure of closeness is required. Based on the nature of the variables (e.g., discrete, continuous or binary) or scales of measurement (e.g., nominal, ordinal, interval or ratio), the choice of proximity measure varies. Some proximity measures are discussed next.

1.2.2 Proximity Measures

In general, clustering requires a proximity measure for discovering similar or dissimilar objects in a dataset. We therefore need to use a measure for distances between objects so that similar objects are a short distance apart and dissimilar ones are further from each other.

Distance Measure

An important step in clustering is to select a distance measure, which determines how the similarity of two elements is calculated. This influences the shape of the clusters, as some elements may be close to one another according to one distance and further away according to another. A metric or distance function defines a distance between elements of a set. The properties of an effective distance measure are the following.

i) $d_{x,y} \geq 0$ (non-negativity): Distance is always positive or zero.

ii) $d_{x,y} = 0$ iff $x = y$ (identity of indiscernibles): Distance is zero if and only if it measured from an object to itself.

iii) $d_{x,y} = d_{y,x}$ (symmetry): Distance is symmetric.

iv) $d_{x,z} \leq d_{x,y} + d_{y,z}$ (triangle inequality): Distances satisfy triangular inequality.

where d is a distance function and x, y, z are objects. A distance function is also called metric if it satisfies all four conditions given above. Because of the triangular inequality (condition 4), not all distance measures are metric, but all metrics are distances. Some of the common distance functions are Euclidean distance (also called 2-norm distance or squared Euclidean distance), Manhattan distance (aka taxicab norm or 1-norm), maximum norm (aka infinity norm), Mahalanobis distance and Hamming distance.

Similarity Measures

From scientific and mathematical points of view, distance is defined as a quantitative degree of how far apart two objects are. Similarity is a numerical quantity that reflects the strength of relationship between two objects or two features. Similarities are higher for pairs of objects that are more alike. This quantity is usually in the range of either -1 to +1 or is normalized to the range from 0 to 1. If the similarity between object x and object y is denoted by $S_{x,y}$, we can measure this quantity in several ways depending on the scale of measurement (or data type). The triangle equality does not hold for similarity measures but the following properties hold true.

i) $S_{x,y} = 1$ only if $x = y$ $(0 \leq S \leq 1)$.

ii) $S_{x,y} = S_{y,x}$ for all x and y (Symmetry).

Some of the common similarity measures are Pearson's correlation, Simple Matching Coefficient, Jaccard Coefficient, Jackknife correlation, Spearman's rank-order correlation coefficient, and Cosine Similarity. There are different types of proximity measures, some of which will be discussed in the next chapter. However, not all the measures are applicable in all domains. The efficiency of the proximity measures are dependent on the domain in which they are applied.

1.2.3 Types of Clustering

Clustering methods are broadly classified into the following categories [HK06, TSK09]:

Partitioning methods

Partitional clustering obtains a partition of a set of objects into k clusters such that each cluster contains at least one object and each object belongs to exactly one cluster. Partitional algorithms typically determine all clusters at once. This type of clustering works well for spherical-shaped clusters. The main disadvantages with this type of clustering are: (i) the number of clusters must be known prior to clustering, (ii) the method detects clusters of spherical shapes only, (iii) detected clusters tend to become uniformly sized and (iv) all objects are forced to belong to a cluster.

Hierarchical methods

Hierarchical clustering is a set of nested clusters that are organized as a tree called "dendrogram". These algorithms can be either agglomerative ("bottom-up") or divisive ("top-down"). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. The general criterion of a good partition is that objects in the same cluster (partition) are more similar to each other than they are to objects in other clusters. Detection of arbitrary shaped clusters is possible using some hierarchical clustering algorithms. This type of clustering suffers from the fact that once a step (merge or split) is done, it cannot be undone. These algorithms are computationally expensive. Some hierarchical algorithms exhibit "chaining effect"– a few objects located so as to form a bridge between two clusters cause objects across the clusters to be grouped into a single elongated cluster.

Density-based methods

The concept of a cluster in density-based clustering is a dense region of objects surrounded by a region of low or no density. Density is the number of objects in the particular neighborhood of a data object. Density-based clustering algorithms are devised to discover arbitrary-shaped clusters. Moreover, they can effectively separate noise and outliers.

Grid-based methods

Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All clustering operations are performed on the grid structure. The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space. The problems inherent with this approach are that the grids are square or rectangular and do not necessarily fit the shape of the clusters. This can be handled by increasing the number of grid cells, but at the expense of computational cost.

Model-based methods

Model-based methods hypothesize a probabilistic model for each of the clusters and find the best fit of the data to the given model. A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based on standard statistics.

Graph-based methods

In graph-based clustering algorithms, graphs are built as combinations of objects, features or both, as nodes and edges. The graph is then partitioned by using graph theoretic algorithms. Graph based clustering algorithms are suitable for data that do not follow a Gaussian or spherical distribution. They can be used to detect clusters of varying shapes and sizes without the need to specify the number of clusters apriori. Graph theoretic algorithms are also used for the problem of clustering cDNAs based on their oligo-nucleotide fingerprints [HSL+99].

Soft Computing

Soft computing differs from conventional (hard) computing in that, unlike hard computing, it deals with imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost. Components of soft comput-

ing include: Fuzzy Systems, Neural Networks, Evolutionary Computation, Machine Learning and Probabilistic Reasoning. Soft computing techniques have been widely used in clustering data where the data deals with uncertainty, overlapping or mixed clusters.

Distributed Clustering

Distributed Data Mining makes the assumption that either the computation or the data itself is distributed. It can be used in environments ranging from parallel supercomputers to P2P networks. It can be applied in areas like distributed information retrieval and sensor networks. Usually distributed clustering algorithms work by first computing a local model. Next, the local models are aggregated by a central node (or a super-peer in P2P clustering systems) and finally either a global model is computed, or aggregated models are sent back to all the nodes to produce locally optimized clusters.

Subspace Clustering

A subspace is a subset of a vector space that is itself a vector space. Subspace clustering is the task of detecting all clusters in all subspaces. Subspace clustering algorithms capture clusters formed by a subset of objects across a subset of features.

The choice of clustering algorithm depends both on the type of data and on the particular purpose of the application. Some clustering algorithms integrate the idea of several clustering methods, so that it is sometimes difficult to classify a given algorithm as uniquely belonging to only one clustering category. Clustering algorithms have been used extensively for many applications and some of them are listed below.

1.2.4 Applications of Clustering

Clustering algorithms can be applied in many fields [HK06, TSK09].

1. Image Analysis: The goal of image clustering is to locate objects and boundaries (lines, curves, etc.) in images. It is the process of assigning a label to

every pixel in an image such that pixels with the same label share certain visual characteristics. Image clustering is used to locate tumors and other pathologies for medical image segmentation, measure tissue volumes, disease diagnosis etc.; to locate objects in satellite images (roads, forests, etc.) [Yam98a]; for face recognition and fingerprint recognition.

2. Coherent Pattern Identification: Clustering techniques have been used to identify the co-expressed genes and coherent patterns from gene expression data. Coherent patterns arise from co-expressed genes and co-expressed genes indicate that the genes are correlated and may participate in similar biological functions [JTZ04]. Gene expression data consists of thousands of genes and identification of co-expressed genes from such data requires use of data mining techniques such as clustering. This helps biologists gain knowledge of similarly expressed genes [Ste06].

3. Marketing: Finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records [HK06].

4. Fraud Detection and Management: Widely used in health care, retail, credit card services, telecommunications (phone card fraud), etc. Such approaches use historical data to build models of fraudulent behavior and use data mining to help identify similar instances [TSK09].

5. Insurance: Identifying groups of motor insurance policy holders with a high average claim cost; detecting a group of people who stage accidents to collect the insurance, detecting professional patients, and detecting rings of doctors and rings of patient references.

6. Document Clustering: Finding groups of documents that are similar to each other based on the important terms appearing in them, identifying frequently occurring terms in each document [TSK09], clustering weblog data to discover groups of similar access patterns.

7. Astronomy: Discovering new galaxies and stars. (The Jet Propulsion Laboratory and the Palomar Observatory discovered 22 quasars with the help of data

mining).

1.3 Discussion

Based on a comprehensive literature survey we come to the following conclusions.

- Clustering algorithms are dependent on the proximity measures used. Choosing an appropriate proximity measure is of utmost importance. That there exists no particular measure which can handle all the issues of clustering further complicates the job. It is highly desirable that the proximity measure used is robust to outliers and can detect clusters inherent in the dataset.

- Various clustering algorithms require different types of input parameters and clustering results are highly dependent on the values of the parameters. Moreover it is difficult to identify different types of clusters in a given dataset. Detection of multi-density and intrinsic or embedded clusters is of utmost importance in identifying the inner structure of a given dataset.

- Handling of massive data can be done efficiently using distributed or parallel clustering techniques.

- Clustering has been applied to many real-world domains. We have studied the application of two real-life domains in this work: satellite imagery and gene expression data.

- Satellite image data contains huge amount of data as well as there is the added problem of mixed pixels. It is highly desirable to solve these problems if the clusters from satellite images are properly identified.

- Gene expression data contain highly connected clusters. Therefore, it would be very helpful if sub-clusters in the dataset can be identified. The sub-clusters consists of genes with highly coherent patterns.

- Due to the large number of microarray experiments being conducted the quantity of gene expression data is always increasing. New genes and new relationships among genes are continuously being discovered. As a result, it is desirable

to cluster the newly available data incrementally instead of having to re-cluster the whole database with every update.

1.4 Contributions of this Thesis

The main contributions of the work reported in this thesis are given in the following subsections chapter wise.

1.4.1 Grid-Density based Clustering for Spatial Data

In Chapter 3, we propose a technique called GDCT, Grid-based Density Clustering using Triangle-subdivision, that can identify arbitrarily shaped embedded clusters as well as multi-density clusters in the presence of noise in large spatial datasets. The triangle-subdivision procedure included in GDCT gives high quality clusters. GDCT clusters the dataset according to the structure of the embedding space. The method exploits a grid based technique to group the data points into blocks and the density of each grid cell is calculated. The blocks are then clustered by a topological search algorithm. For finer clustering, the triangle-subdivision method is used. The technique finds quality clustering over variable density space. The experiments establish that GDCT is superior than other comparable algorithms in terms of cluster quality.

To evaluate the technique in terms of clustering quality, we use several synthetic datasets as well as the Chameleon datasets [KHK99]. From our experimental results, we conclude that GDCT is highly competent in detecting intrinsic as well as multi-density clusters.

1.4.2 Grid-Density based Clustering for Pan-Chromatic and Multi-Spectral Satellite Data

Chapter 4 presents two grid-density based clustering techniques for satellite data.

The first technique, SATCLUS, works in two phases: Phase I which identifies coarse

or rough clusters and Phase II which smoothens the cluster boundaries detected in Phase I. It can handle the detection of irregular shaped clusters by pixel level processing of the cluster borders (border smoothening process during Phase II). In Phase II, the cluster boundaries detected in Phase I are smoothened by incorporating a partitioning approach. The second technique, GDSDC, has the same first phase as SATCLUS, while in the second phase a fuzzy approach is used to improve the correctness of the cluster borders and detect the mixed pixels present in the dataset. Both SATCLUS and GDSDC do not require specification of the initial cluster centers. Neither does the number of clusters play any role in the clustering process. The proposed techniques were tested on a large number of multi-spectral satellite imagery and the cluster results are found to be of excellent quality. A major advantage of this method is its simplicity and being free of the need to make initial guesses about cluster centers or the number of clusters.

Experimental results establish that SATCLUS and GDSDC can detect all classes present in any satellite data effectively and dynamically. Moreover, the clusters of the remotely sensed multi-spectral satellite images obtained by SATCLUS were validated by using an index β as in [PGS00] as well as the homogeneity measure [SMKS03].

1.4.3 Distributed Grid-Density based Clustering

In Chapter 5, we present two distributed clustering techniques for handling massive datasets, one in spatial domain and the other for satellite imagery.

The first technique can handle voluminous data and at the same time effectively detect multiple nested or embedded clusters even in presence of noise. The distributed grid-density based clustering technique (DGDCT) finds clusters in spatial datasets according to the structure of the embedding space and can address the scalability problem effectively. Better speedup and scale-up is the major attraction of the proposed technique. DGDCT can detect global as well as embedded clusters by sharing the computational efforts among k_p processors. Experimental results using several synthetic and the Chameleon datasets [KHK99] establish the superiority of

the technique. Scale-up and speedup results establish the superiority of the technique using several synthetic datasets.

The second distributed clustering technique (DisClus) can handle high resolution satellite datasets qualitatively. The method exploits a grid-based technique to group the data points into blocks and to calculate the density of each grid cell. The blocks are then clustered by a neighborhood search algorithm. Finally, the pixels of the border cells of the clusters already detected are reassigned to the clusters using a partitioning based approach to get finer results.

While comparing DisClus to several competitive algorithms, we find that DisClus requires only two parameters: the number of grid cells $gr_n \times gr_n$ and a merging threshold. However, we observe in our experiments that the merging threshold does not vary significantly with different datasets.

Experimental results establish the capability of DisClus to handle large satellite image data and to determine all clusters present effectively and dynamically. DisClus is also superior in terms of execution time as well as relative speedup.

1.4.4 Clustering Gene Expression Data for Coherent Pattern Identification

In Chapter 6, two clustering techniques capable of identifying coherent patterns in gene expression data are presented.

The first, GenClus, is a technique for clustering gene expression datasets, designed based on a density based approach. It can identify clusters and sub-clusters of arbitrary shapes in any gene expression dataset in presence of noise. Another advantage of this technique is that it retains the magnitude as well as the regulation information. It uses no proximity measures and is therefore free from the limitations imposed by them. To test the performance of the clustering technique, we compare the clusters identified by our method with the results from k-means, SOM, DCCA

and RDClust and find it to be superior.

Later in this chapter, we present an incremental version of GenClus (InGenClus) that is capable of handling datasets that are updated incrementally. InGenClus was tested using various datasets. The clusters obtained by InGenClus using the dataset from [CCW+98] are same as those obtained by GenClus.

The second technique presents an effective tree-based clustering technique (GeneClusTree) for finding clusters in gene expression data. GeneClusTree finds all clusters in subspaces using a tree-based density approach by scanning the whole database in the minimum number of possible passes. Another important advantage of GeneClusTree is that it does not use a proximity measure. Our technique works by finding the maximal space clusters and then proceeds in finding the reduced space clusters. The clusters are represented as a tree with the reduced space clusters as the child of its respective maximal space cluster. Effectiveness of GeneClusTree is established in terms of well known z-score measure and p-value over several real-life datasets. The datasets have been obtained from http://faculty.washington.edu/kayee/cluster. GeneClusTree was compared with UPGMA, RDClust and DCCA w.r.t. z-score and found to give better result. The p-value analysis of GeneClusTree shows that it is capable in detecting biologically relevant clusters from gene expression data.

We note here that both of our methods, GenClus and GeneClusTree, do not require the number of clusters as an input parameter. They detect the clusters present in the dataset automatically and gives the rest as noise.

1.5 Organization of the Thesis

The thesis is organized as follows:

- *Chapter* 2 gives a survey of literature regarding different clustering and outlier detection techniques.

- *Chapter* 3 presents our own grid-density based clustering technique with triangle subdivision (GDCT) for spatial data. The measure is established to be appropriate in detecting variable density and embedded clusters with respect to different synthetic datasets.

- In *Chapter* 4, two Grid-Density based clustering techniques for pan-chromatic and multi-spectral satellite data is presented.

- Two distributed clustering techniques for handling massive datasets, one in spatial domain (SATCLUS) and the other for satellite imagery (DisClus) is reported in *Chapter* 5.

- *Chapter* 6 of this thesis describes two gene expression clustering (GenClus and GeneClusTree). An incremental version of GenClus algorithm that can handle incremental datasets is also presented in *Chapter* 6.

- Finally, concluding remarks and future works are given in *Chapter* 7.

The next chapter presents a selected survey of various clustering methods and outlier detection techniques.

Chapter 2

Related Work

Cluster analysis is the process of division of data into groups (clusters) that are meaningful, useful, or both. It should reflect the natural structure of the data. Cluster analysis groups data objects based on information present in the data. The goal of clustering is to group similar or related objects in the same cluster and different or unrelated groups in different clusters. Clustering objects into meaningful groups is based on similarity or dissimilarity measures. Cluster analysis is a difficult problem because of many factors such as effective similarity measures, criterion functions, algorithms and initial conditions. Moreover, it is well known that no clustering method can adequately handle all sorts of cluster structures (shape, size and density). Outlier detection is one of the major technologies in data mining, whose task is to find small groups of data objects that are considerably different from rest of the data. Outlier mining is applied over various fields such as telecommunication, financial fraud detection, and data cleaning. In outlier mining, the patterns lying behind the outliers are usually interesting and helps the decision makers to make profit or to improve the service quality. It is considered to be an important research area, and outlier detection is studied intensively by the data mining community [BKNS00, HK06]. In the succeeding sections, we discuss some commonly used proximity measures, various well-known clustering techniques and also some of the existing outlier detection techniques are discussed.

2.1 Proximity Measures

From scientific and mathematical point of view, distance is defined as a quantitative degree of how far apart two objects are. Similarity is a numerical quantity that reflects the strength of relationship between two objects or two features. Similarities are higher for pairs of objects that are more alike. This quantity is usually in the range of either -1 to +1 or is normalized into 0 to 1. If the similarity between a pair of objects (x, y) is denoted by $S_{x,y}$, we can measure this quantity in several ways depending on the scale of measurement (or data type) that we have.

Distance measure is also known as dissimilarity measure. Similarity and dissimilarity measures are often called proximity measures. Dissimilarity measures the discrepancy between the two objects, i.e., it measures the degree to which two objects are different. There are many types of distance and similarity measures. Each similarity or dissimilarity measure has its own characteristics. Next, we consider several important issues concerning proximity measures.

2.1.1 Relationship between Similarity and Dissimilarity

Let normalized dissimilarity between object x and object y be denoted by $d_{x,y}$. Then the relationship between dissimilarity and similarity [HK06] is given by

$$S_{x,y} = 1 - d_{x,y}. \tag{2.1}$$

Here, $S_{x,y}$ is normalized similarity between objects x and y. Similarity is bounded by 0 and 1. When similarity is one (i.e., two objects are exactly similar), the dissimilarity is zero and when the similarity is zero (i.e., two objects are very different), dissimilarity is one. If the value of similarity has range of -1 to +1, and the dissimilarity is measured with range of 0 and 1, then

$$S_{x,y} = 1 - 2d_{x,y}. \tag{2.2}$$

When dissimilarity is one (i.e., two objects are very different), similarity is minus one and when the dissimilarity is zero (i.e., two objects are very similar), similarity is one. In many cases, measuring dissimilarity (i.e., distance) is easier than measuring

similarity. Once we can measure dissimilarity, we can easily normalize it and convert it to similarity measure. It is also common for dissimilarities to range from 0 to ∞. Frequently, proximity measures are transformed to the interval [0, 1]. The transformation of similarities to the interval [0, 1] is given by

$$S'_{x,y} = \frac{S_{x,y} - min\_S_{x,y}}{max\_S_{x,y} - min\_S_{x,y}} \tag{2.3}$$

where, $min\_S_{x,y}$ and $max\_S_{x,y}$ are minimum and maximum similarities respectively. Similarly, dissimilarity measures with a finite range can be mapped to the interval [0, 1] by using the formula

$$d'_{x,y} = \frac{d_{x,y} - min\_d_{x,y}}{max\_d_{x,y} - min\_d_{x,y}} \tag{2.4}$$

where, $min\_d_{x,y}$ and $max\_d_{x,y}$ are minimum and maximum dissimilarities respectively.

If the proximity measure has values in the range $[0, \infty]$, then a non-linear transformation is needed and the values in the transformed scale will not have the same relationship to one another as the original. But, whether such a transformation is desirable or not depends on the application it is used.

2.1.2 Some Distance Measures

A popular distance measure based on variables that take on continuous values is to standardize the values by dividing by the standard deviation (sometimes other measures such as range are used) and then to compute the distance between objects using the Euclidean metric.

The *Euclidean distance* $d_{i,j}$ between two objects, i and j with variable values $(x_{i1}, x_{i2}, \cdots, x_{in})$ and $(x_{j1}, x_{j2}, \cdots, x_{jn})$ is defined by:

$$d_{i,j} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2} \tag{2.5}$$

If some variables should be given more importance than others then the squared difference terms should be multiplied by weights (positive numbers adding up to

one) and use larger weights for the important variables. The *Weighted Euclidean distance* measure is given by:

$$d_{i,j} = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \cdots + w_n(x_{in} - x_{jn})^2} \tag{2.6}$$

where $w_1, w_2, \cdots, w_n$ are the weights for variables $1, 2, \cdots, n$ so that $w_i \geq 0$, $\sum_{i=1}^{n} w_i = 1$.

Other useful measures of dissimilarity other than the Euclidean distance that satisfy the triangular inequality and so qualify as distance metrics are:

Manhattan distance is defined by

$$d_{i,j} = \sum_{m=1}^{n} \mid x_{im} - x_{jm} \mid \tag{2.7}$$

Minkowski distance is the generalized form of the two distance metrics discussed above. It is given as

$$d_{i,j} = \sqrt[p]{\sum_{m=1}^{n} |x_{im} - x_{jm}|^p} \tag{2.8}$$

where p is a parameter. For $p = 1$, we get the Manhattan distance, and for $p = 2$ we get the Euclidean distance.

Mahalanobis distance corrects data for different scales and correlations in the variables. It is defined by

$$d_{i,j} = \sqrt{(x_i - x_j)'V'(x_i - x_j)} \tag{2.9}$$

where x_i and x_j are n-dimensional vectors of the variable values for i and j respectively; and V is the covariance matrix for these vectors. This measure takes into account the correlation between the variable: variables that are highly correlated with other variables do not contribute as much as variables that are uncorrelated or mildly correlated.

Maximum co-ordinate distance is defined by

$$d_{i,j} = max_{m=1,2,\ \ ,n}|x_{im} - x_{jm}| \tag{2.10}$$

For $p = \infty$ we get the *Chebyshev distance* (L_{max} or L_∞ norm) named after Chebyshev [HDRT04]. This is the maximum distance between any pair of attributes of the objects. Formally, L_∞ is defined as

$$d_{i,j} = \lim_{p \to \infty} \left(\sum_{m=1}^{n} |x_{im} - x_{jm}|^p \right)^{\frac{1}{p}}. \tag{2.11}$$

Hamming distance [Ham50] between two strings of equal length is the number of positions at which the corresponding symbols are different. It measures the minimum number of substitutions required to change one member into another. For a fixed length n, the Hamming distance is a metric on the vector space of the words of that length, as it obviously fulfills the conditions of non-negativity, identity of indiscernibles and symmetry, and it can be shown easily by complete induction that it satisfies the triangle inequality as well. For binary strings i and j the Hamming distance is equal to the number of ones in i XOR j. If q_{10} = number of variables with value 1 for the i^{th} object and 0 for the j^{th} object and q_{01} = number of variables with value 0 for the i^{th} object and 1 for the j^{th} object, we have

$$d_{i,j} = q_{10} + q_{01} \tag{2.12}$$

2.1.3 Some Similarity Measures

Sometimes it is more natural or convenient to work with a similarity measure between objects rather than distance which measures dissimilarity. Such measures can always be converted to distance measures. In the above example we could define a distance measure $d_{i,j} = 1 - S_{i,j}$.

Pearson's correlation: The correlation coefficient, $\rho_{i,j}$ is a widely used similarity measure, defined by

$$\rho_{i,j} = \sqrt{\frac{\sum_{m=1}^{n}(x_{im} - \overline{x_i})(x_{jm} - \overline{x_j})}{\sqrt{\sum_{m=1}^{n}((x_{im} - \overline{x_i}))^2 \sum_{m=1}^{n}((x_{jm} - \overline{x_j}))^2}}} \tag{2.13}$$

where, $\overline{x_i}$ is the mean of the n attributes of the i^{th} object and $\overline{x_j}$ is the mean of the n attributes of the j^{th} object and Similarity measures between objects that have only

binary attributes are called *similarity coefficients* and have values between 0 and 1 A value of 0 means that the objects are completely dissimilar and a value of 1 means that the objects are completely similar

Suppose objects i and j have n binary attributes Then, on comparing i and j the following quantities are obtained

i) q_{00} the number of attributes where $i = 0$ and $j = 0$,

ii) q_{01} the number of attributes where $i = 0$ and $j = 1$,

iii) q_{10} the number of attributes where $i = 1$ and $j = 0$, and

iv) q_{11} the number of attributes where $i = 1$ and $j = 1$

Using the above quantities different similarity coefficients can be obtained

Simple Matching Coefficient or SMC [HK06] is one of the most commonly used similarity coefficients and is defined as,

$$SMC = \frac{Total\ number\ of\ matched\ attributes}{Total\ attributes} = \frac{q_{00} + q_{11}}{q_{00} + q_{01} + q_{10} + q_{11}} \quad (2\ 14)$$

SMC gives equal weight to both presences and absences

Jaccard Coefficient [HK06] is used for handling objects consisting of asymmetric binary attributes Jaccard Coefficient (J) is defined as follows,

$$J = \frac{Number\ of\ matched\ attributes}{Total\ attributes - non\ existence\ of\ both\ attributes} = \frac{q_{11}}{q_{01} + q_{10} + q_{11}} \quad (2\ 15)$$

Pearson's correlation is a powerful similarity measure However, empirical study has shown that it is not robust with respect to outliers [HKY99], thus potentially yielding false positives which assign a high similarity score to a pair of dissimilar patterns If two patterns have a common peak or valley at a single feature, the correlation will be dominated by this feature, although the patterns at the remaining features may be completely dissimilar Another drawback of Pearson's correlation coefficient is that it assumes an approximate Gaussian distribution of the points and may not be

robust for non-Gaussian distributions [Bic01].

Jackknife correlation [JTZ04], helps in overcoming the single outlier problem of Pearson's correlation. It is defined as,

$$Jacknife_{i,j} = min\{\rho_{i,j}^1, \cdots, \rho_{i,j}^l, \cdots, \rho_{i,j}^n\} \tag{2.16}$$

where $\rho_{i,j}^l$ is the Pearson's correlation coefficient of data objects i and j with the l^{th} feature deleted. Use of Jackknife correlation avoids the "dominance effect" of single outliers. More general versions of Jackknife correlation that are robust to more than one outlier can similarly be derived. However, generalized Jackknife correlation, which involves the enumeration of different combinations of features to be deleted, is computationally costly and is rarely used.

Spearman's rank-order correlation coefficient is used to address the problem of non-Gaussian distributions w.r.t. Pearson's correlation, the Spearman's rank-order correlation coefficient [JTZ04] has been suggested as a similarity measure. The ranking correlation is derived by replacing the data x_{in} with its rank r_{in} among all conditions. For example, $r_{in} = 3$ if x_{in} is the third highest value among x_{if}, where $1 \leq f \leq n$. Spearman's correlation coefficient does not require the assumption of Gaussian distribution and is more robust against outliers than Pearson's correlation coefficient. However, as a consequence of ranking, a significant amount of information present in the data is lost.

Cosine Similarity [TSK09] is useful for finding document similarity. If x and y are two document vectors, then $cos_{x,y}$ is given by the following equation,

$$cos_{x,y} = \frac{x \cdot y}{\|x\|\|y\|} \tag{2.17}$$

where . indicates the vector dot product, $x.y = \sum_{k=1}^{n} x_k y_k$, and $\|x\|$ is the length of vector x, and $\|x\| = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{x \cdot x}$.

CorHsim: In [LWN+09], a new similarity measure for gene expression microarray data, *CorHsim*, is presented. It reflects the magnitude and shape information

of gene expression data at the same time and is defined as follows:

$$CorHsim_{x,y} = \frac{1}{2n}\sum_{i=1}^{n}\left(\frac{\mid (x_i - \overline{x})(y_i - \overline{y}) \mid}{\sigma_x \sigma_y} + \frac{1}{1+ \mid x_i - y_i \mid}\right) \tag{2.18}$$

where, σ_x and σ_y are the standard deviations of x and y respectively. The disadvantage of $CorHsim$ is that it uses the mean value and may sometimes represent the pattern differently.

The similarity/dissimilarity measures discussed above have been applied in various domains. However, not all the measures are applicable throughout all domains. There is a qualitative domain specific dependency among similarity/dissimilarity measures.

2.2 Existing Clustering Approaches

Generally, clustering algorithms are categorized into partitioning methods, hierarchical methods, density-based methods, grid-based methods, model-based methods, graph based methods, cluster ensembles, distributed methods, soft computing methods, and subspace clustering [JMF99].

2.2.1 Partitional

Partitional clustering divides the set of data objects into non-overlapping (disjoint) clusters such that each object is in exactly one cluster. Partitioning methods are divided into two major subcategories [HK06], the centroid based and the medoid based algorithms. The centroid based algorithms represent each cluster by using the gravity center (mean) of the instances while the medoid algorithms represent each cluster by means of the instances closest to the mean.

The k-means algorithm is one of the most well-known centroid algorithm. The k-means method partitions the dataset into k subsets such that all points in a given subset are closest to the same center. In randomly selects k of the instances to represent the cluster centers and based on the selected attributes, all remaining instances

are assigned to their nearest cluster center. K-means then computes the new cluster centers by taking the mean of all data points belonging to the same cluster. The process is iterated until some convergence criterion is met (usually till there is no change in the cluster centers). Generally, the k-means algorithm has the following important properties: (i) It is efficient in processing large datasets, (ii) It often terminates at a local optimum, (iii) The clusters have spherical shapes, (iv) It is sensitive to noise. However, the number of clusters have to be provided as an input parameter and choosing the proper initial centroids is the key step of the basic k-means procedure and results are dependent on it.

K-medoid is also a partitional clustering technique that clusters the dataset of N objects into k clusters and is more robust to noise and outliers as compared to k-means. A medoid is the most centrally located object in a given dataset. It can be defined as that object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal[HK06]. One of the most popular k-medoid clustering is the Partitioning around Medoids (PAM) algorithm which begins with an arbitrary set of k objects as medoid points out of N data objects. Each data object in the given dataset is associated to the most similar medoid. Then a non-medoid object, say o_i, is selected randomly and the total cost S_{cost} of swapping initial medoid object to o_i is computed. If $S_{cost} < 0$, then swap initial medoid with the new one. The process of selection of medoids and swapping is iterated until there is no change in the medoid.

The k-modes algorithm [Hua98] is a recent partitioning algorithm and uses the simple matching coefficient measure to deal with categorical attributes. For clustering instances described by mixed attributes, a k-prototypes algorithm [Hua98] is also proposed that integrates the k-means and k-modes algorithms and uses a combined dissimilarity measure during clustering. In [Che03], a generalization of conventional k-means clustering algorithm has been presented that is applicable to ellipse-shaped data clusters as well as ball-shaped ones and does not require the exact cluster number apriori.

2.2.2 Hierarchical

Hierarchical clustering [HK06, TSK09] provides a nested sequence of partitions, represented graphically with a *dendrogram*. Each node (cluster) in the tree (except the leaf nodes) is the union of its children (sub-clusters), and the root of the tree is the cluster containing all the objects. Sometimes the leaf nodes consists of a single object and are termed as singleton clusters. Hierarchical methods are divided into two major subcategories: (i) agglomerative method, which forms the clusters in a bottom-up fashion starting with each object in a separate cluster and merging them until all data instances belong to the same cluster and (ii) divisive method, which splits up the dataset into smaller clusters in a top-down fashion until each cluster contains only one instance. Both divisive algorithms and agglomerative algorithms can be represented by dendrograms and are known for their quick termination. Other merits include: (a) they do not require the number of clusters to be known in advance, (b) they compute a complete hierarchy of clusters, (c) good result visualizations are integrated into the methods, and (d) a flat partition can be derived later by cutting through the dendrogram. However, both methods suffer from their inability to perform adjustments once the splitting or merging decision is made. Hierarchical clustering techniques use various criteria to decide locally at each step which clusters should be merged (or split for divisive approaches). To merge or split clusters, the distance between individual objects has been generalized to the distance between subsets. Such derived proximity measure is called a linkage metric. Major inter-cluster linkage includes: single-link, complete-link and average-link [JMF99]. The single-link similarity between two clusters is the similarity between the two most similar instances, one of which appears in each cluster. Single link is good at handling non-elliptical shapes, but is sensitive to noise and outliers. The complete-link similarity is the similarity between the two most dissimilar instances, one from each cluster. Complete link is less susceptible to noise and outliers, but can break large clusters, and has trouble with convex shapes. The average-link similarity is a compromise between the two. There are various hierarchical clustering algorithms, some of them are: Balanced Iterative Reducing and Clustering using Hierarchies BIRCH [ZRL96], Clustering Using REpresentatives CURE [GRS98] and CHAMELEON [KHK99].

BIRCH [ZRL96] creates a height-balanced tree of nodes that summarize data by accumulating its zero, first, and second moments (CF statistics). It uses a hierarchical data structure called CF-tree for partitioning the incoming data objects in an incremental and dynamic way. CF-tree is a height-balanced tree, which stores the clustering features and it is based on two parameters: branching factor Bf and threshold Th, which refer to the diameter of a cluster (the diameter (or radius) of each cluster must be less than Th). A CF tree is built as the data is scanned. While inserting each data object, the CF tree is traversed, starting from the root and choosing the closest node at each level. When the closest leaf cluster for the current data object is finally identified, a test is performed to see if adding the data object to the candidate cluster will result in a new cluster with a diameter greater than the given threshold, Th. If it fits the leaf well and if the leaf is not overcrowded, CF statistics are incremented for all nodes from the leaf to the root. Otherwise a new CF is constructed. Since the maximum number of children per node (Bf) is limited, one or several splits can happen. When the tree reaches the assigned memory size, it is rebuilt and Th is updated to a coarser one. The outliers are sent to disk, and refitted gradually during tree rebuilds. BIRCH can typically find a good clustering with a single scan of the data and improve the quality further with a few additional scans. It can also handle noise effectively. Moreover, because BIRCH is reasonably fast ($O(N)$), it can be used as a more intelligent alternative to data sampling in order to improve the scalability of other clustering algorithms. However, it may not work well when clusters are not spherical because it uses the concept of radius or diameter to control the boundary of a cluster. In addition, it is order-sensitive as it may generate different clusters for different orders of the same input data. Bubble and Bubble-FM [GRG+98] clustering algorithms are extensions of BIRCH to handle categorical data.

In CURE [GRS98], multiple well-scattered objects (representative points) are chosen to represent a cluster. These points usually capture the geometry and shape of the cluster. The first representative point is chosen to be the point farthest from the cluster center, while the remaining points are chosen so that they are farthest from all previously chosen points. This ensures that the representative points are naturally

relatively well distributed. The number of points chosen, is a parameter, but it was found that a value of 10 or more worked well. The similarity between two clusters is measured by the similarity of the closest pair of the representative points (after they are shrunk toward their respective centers) belonging to different clusters. Once the representative points are chosen they are shrunk toward the center by a factor which ranges between 0 and 1. This helps moderate the effect of outliers, which are usually farther away from the center and thus are shrunk more. CURE uses an agglomerative hierarchical scheme to perform the actual clustering. Unlike centroid/medoid based methods, CURE is capable of finding clusters of different shapes and sizes, as it represents each cluster via multiple representative points. Shrinking the representative points towards the center helps CURE in avoiding the problem of noise. However, it cannot be applied directly to large datasets. For this reason, CURE takes a random sample and performs the hierarchical clustering on the sampled data points.

ROCK [GRS99], is a clustering algorithm for categorical data and uses the Jaccard coefficient as a measure of similarity. It uses the concept of links i.e., the number of common neighbors for any two objects. ROCK first draws a random sample from the dataset and then performs clustering of the data with links. Finally the data in the disk is labeled. It accepts as input the sampled set S to be clustered (that are drawn randomly from the original dataset), and the number of desired clusters k. ROCK samples the dataset in the same manner as CURE.

CHAMELEON [KHK99] uses a two-phase approach to cluster the data. In the first phase, it uses a graph partitioning algorithm to divide the dataset into a set of individual clusters. It generates a k-nearest neighbor graph that contains links only between a point and its k-nearest neighbors. During the second phase, it uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly merging these sub-clusters. None of the clusters formed can contain less than a user specific number of instances. Two clusters are merged only if the inter-connectivity and closeness (proximity) between two clusters are high relative to the internal inter-connectivity of the clusters and closeness of items within the clusters. Therefore, it is better than both CURE and ROCK as CURE ignores information

about inter-connectivity of the objects while ROCK ignores information about the closeness of two clusters.

A novel incremental hierarchical clustering algorithm (GRIN) for numerical datasets based on gravity theory in physics is presented in [CHO02]. One main factor that makes the GRIN algorithm able to deliver favorite clustering quality is that the optimal parameters settings in the GRIN algorithm are not sensitive to the distribution of the dataset.

For 2D spatial data (for example, GIS database) the algorithm AMOEBA [ECL00] uses Delaunay diagram (the dual of Voronoi diagram) to represent data proximity and has $O(NlogN)$ complexity. The algorithm consists of two steps: (i) The Delaunay diagram is constructed and a connected planar plane-embedded graph is passed to the algorithm to act as the diagram. Clusters are made up of the points in a connected component and the points in the clusters are reported recursively. Every edge is matched against the criteria and passive edges and noise are discarded; active edges and their points form proximity sub-graphs at each level of the hierarchy. (ii) The algorithm calls itself recursively until no new connected components are created when the passive edges and noise are discarded.

The advantages of hierarchical clustering include: (i) Embedded flexibility regarding the level of granularity, (ii) Ease of handling of any forms of similarity or distance, and (iii) Applicability to any attribute types. The disadvantages of hierarchical clustering are: (i) Vagueness of termination criteria, (ii) The fact that most hierarchical algorithms do not revisit once intermediate clusters that have been constructed with the purpose of their improvement.

2.2.3 Density based

Density-based clustering algorithms try to find clusters based on density of data objects in a region. The key idea of density-based clustering is that for each core object of a cluster the neighborhood of a given radius (ε) has to contain at least a minimum number of instances ($MinPts$), where ε and $MinPts$ are the two input

parameters. One of the most well known density-based clustering algorithms is the DBSCAN [EKSX96]. This algorithm grows regions with sufficiently high density into clusters. DBSCAN separates data objects into three classes as illustrated in Figure 2.1.

- Core points: These points are at the interior of a cluster. A point is an interior point if there are enough points in its neighborhood.
- Border points: A border point is a point that is not a core point, i.e., there are not enough points in its neighborhood, but it falls within the neighborhood of a core point.
- Noise points: A noise point is any point that is not a core point or a border point.

The neighborhood within a radius ε of an object, say p is called the ε-neighborhood of p. If the ε-neighborhood of p contains at least $MinPts$ number of objects then p is a core object. DBSCAN's definition of a cluster is based on the notion of density-reachability. The basic concepts of DBSCAN are directly density-reachability, density-reachability and density connectivity. Basically, an object q is directly density-reachable from an object p if it is within the ε-neighborhood of a core object, p. An object q is called density-reachable from p if there is a sequence of objects $p_1, p_2, \cdots, p_n$ such that $p_1 = p$ and $p_n = q$ where each p_{i+1} is directly density-reachable from p_i. The relation of density-reachability is not symmetric (since q might lie on the edge of a cluster, having insufficient number of neighbors for q to be core). Two objects p and q are density-connected if there is an object o such that both p and q are density-reachable from o. A cluster, can therefore be defined as a subset of the objects of the database that satisfies two properties: (i) all objects within the cluster are mutually density-connected and (ii) if an object is density-connected to any point of the cluster, it is part of the cluster as well. To find a cluster, DBSCAN starts with an arbitrary object (p) in dataset (D) and retrieves all objects of D w.r.t. ε and $MinPts$. DBSCAN has a number of advantages such as detection of arbitrary shaped clusters, noise handling and is hence quite attractive. However, it suffers from huge computational requirements (the time complexity is $O(N)^2$). So it can take huge amounts of time with large datasets. One way to

overcome this problem is to build spatial index structure over the dataset like R^* tree to locate points within ε distance from the core points of the clusters. But this solution is suitable only when the dimensionality of the data is low. Also DBSCAN is dependent on the input parameters ε and $MinPts$ and there is no straight forward way to fit them to the data. Moreover, different parts of data could require different parameters due to variation in density of the parts.

The algorithm OPTICS (Ordering Points To Identify the Clustering Structure) [ABKS99] can detect clusters of variable density by creating an ordering of the dataset that represents its density-based clustering structure. OPTICS considers a minimum radius (ε') that makes a neighborhood legitimate for the algorithm. It is a versatile basis for interactive cluster analysis and is consistent with DBSCAN, but goes a step further by keeping the same two parameters ε, $MinPts$ and introducing the concept of core-distance ε' (distance to $MinPts$ nearest neighbor when it does not exceed ε, or undefined otherwise). OPTICS covers a spectrum of all different $\varepsilon' \leq \varepsilon$. The constructed ordering can be used automatically or interactively. With each point, OPTICS stores only two additional fields, the so-called core- and reachability-distances. Experimentally, OPTICS exhibits runtime roughly equal to 1.6 of DBSCAN runtime. While OPTICS can detect the different local densities, it is highly sensitive to its three parameters.

An incremental version of DBSCAN (incremental DBSCAN) is presented in [EKS+98]. It has been proven that this incremental algorithm yields the same result as DBSCAN. In addition, another clustering algorithm (GDBSCAN) generalizing the density-based algorithm DBSCAN is presented in [SEKX98]. GDBSCAN can be applied to both numerical and categorical attributes. Furthermore, DBCLASD (Distribution Based Clustering of Large Spatial Datasets) eliminates the need for ε and $MinPts$ parameters [XEKS98]. DBCLASD incrementally augments an initial cluster by its neighboring points as long as the nearest neighbor distance set of the resulting cluster still fits the expected distance distribution. DBSCLAD defines a cluster as a non-empty arbitrary shape subset in D that has the expected distribution of distance to the nearest neighbor with a required confidence, and is the maximal connected set

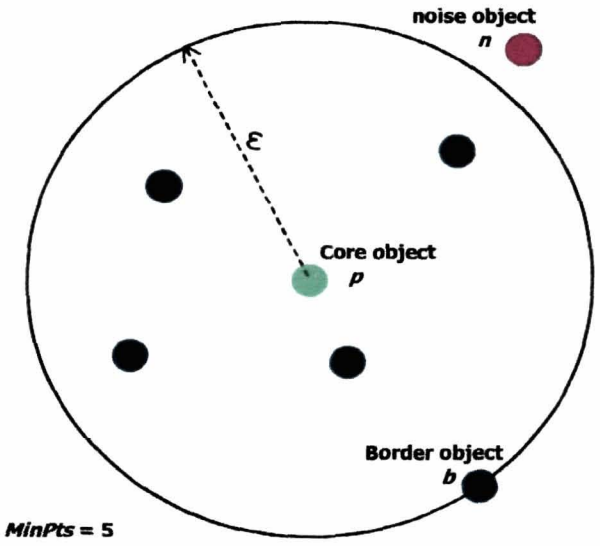


Figure 2.1: Core, border and noise objects in an example dataset

with this quality. Regarding connectivity, DBCLASD relies on grid-based approach to generate cluster-approximating polygons. The algorithm contains devices for handling real databases with noise and implements incremental unsupervised learning and can also handle spatial data. Two concepts are used here. First, assignments are not final: points can change cluster membership. Second, certain points (noise) are not assigned, but are tried later. Therefore, once incrementally fetched points can be revisited internally. DBCLASD is known to run faster than CLARANS by a factor of 60 on some examples. In comparison with much more efficient DBSCAN, it can be 2-3 times slower. However, DBCLASD requires no user input, while empirical search for appropriate parameter requires several DBSCAN runs. In addition, DBCLASD discovers clusters of different densities.

Another density-based algorithm is the DENCLUE [HHK98]. The basic idea of DENCLUE is to model the overall point density analytically as the sum of influence functions of the data points. The influence function can be seen as a function, which describes the impact of a data point within its neighborhood. Then, by determining the maximum of the overall density function it can identify the clusters present. The algorithm allows a compact mathematical description of arbitrarily shaped clusters

in high-dimensional datasets and is significantly faster than the other density based clustering algorithms. Moreover, DENCLUE produces good clustering results even when a large amount of noise is present. As in most other approaches, the quality of the resulting clustering depends on an adequate choice of the parameters. In this approach, there are two important parameters, the parameter σ_1 determines the influence of a point in its neighborhood and η_1 describes whether a density-attractor is significant. Density-attractors are local maxima of the overall density function. The runtime of DENCLUE scales with N sub-linearly. This is due to the fact that though all the points are fetched, the bulk of analysis (in clustering stage) involves only points in highly populated areas.

In [JPZ03], the Density-based Hierarchical Clustering method (DHC) is presented. It considers a cluster as a high-dimensional dense area, where data objects are attracted to each other. At the core part of the dense area, objects have higher density whereas objects at the peripheral area are relatively sparse. Once the density and attraction of data objects are defined, DHC organizes the cluster structure of the dataset in two-level hierarchical structures: attraction tree (represents relationships of objects in the dense area) and density tree (summarizes the cluster structure of the attraction tree where each node represents a dense area). Density tree is mined (split into sub-dense areas based on some criteria) for the final clusters. DHC is effective for the high-connectivity characteristic of gene expression data because it first captures the core part of the cluster and then divides the borders of the clusters on the basis of the attraction between the data objects. The two-level hierarchical representation of the dataset enables the relationship among the clusters and also organizes the relationship among the data objects within the same cluster. The computational complexity of this step is $O(N^2)$, which makes DHC inefficient. Furthermore, two global parameters used in DHC to control the splitting process of dense areas are also sensitive.

FDC algorithm (Fast Density-Based Clustering) is presented in [ZCK99] for density-based clustering defined by the density-linked relationship. The clustering in this algorithm is defined by an equivalence relationship on the objects in the database.

The complexity of FDC is linear to the size of the database, which is much faster than that of the algorithm DBSCAN. More recently, the algorithm SNN (Shared Nearest Neighbors) [ESK03] blends a density based approach with the idea of ROCK. SNN sparsifies similarity matrix by only keeping k-nearest neighbors, and thus derives the total strength of links for each object.

2.2.4 Grid based

Grid-based clustering algorithms first quantize the clustering space into a finite number of cells (hyper-rectangles) and then perform the required operations on the quantized space. Cells that contain more than certain number of points are treated as dense and the dense cells are connected to form the clusters. Here, we report some of the grid-based clustering algorithms such as STatistical INformation Grid-based method - STING [WYM97], WaveCluster [SC+98], and CLustering In QUEst - CLIQUE [AGGR98].

STING [WYM97] is a grid based multi resolution clustering technique in which the spatial area is divided into rectangular cells in order to form a hierarchical structure. The cells in a high level are composed from the cells in the lower level. Each cell has four (default) children and stores a point count, and attribute-dependent measures: mean, standard deviation, minimum, maximum, and distribution type. Measures are accumulated starting from bottom level cells, and further propagate to higher-level cells (e.g., minimum is equal to a minimum among the children-minimums). It generates a hierarchical structure of the grid cells so as to represent the clustering information at different levels. Therefore, STING constructs data summaries and assembles statistics in a hierarchical tree of nodes that are grid-cells. Although STING generates good clustering results in a short running time, there are two major problems with this algorithm. Firstly, the performance of STING relies on the granularity of the lowest level of the grid structure. Secondly, the resulting clusters are all bounded horizontally or vertically, but never diagonally. This shortcoming might greatly affect the cluster quality.

CLIQUE [AGGR98] uses the concepts of density and grid based methods. CLIQUE

starts by finding all the dense areas in the one-dimensional spaces corresponding to each attribute. CLIQUE then generates the set of two-dimensional cells that might possibly be dense, by looking at dense one-dimensional cells, as each two-dimensional cell must be associated with a pair of dense one-dimensional cells. The dense units are then connected to form clusters. It uses apriori algorithm (bottom up algorithm) to find dense units. Generally, CLIQUE generates the possible set of n-dimensional cells that might possibly be dense by looking at dense $(n-1)$ dimensional cells. CLIQUE is able to find clusters in all subspaces of the original data space and present a minimal description of each cluster in the form of a DNF expression. Steps involved in CLIQUE is i) identification of subspaces (dense Units) that contain cluster ii) merging of dense units to form cluster and iii) Generation of minimal description for the clusters. CLIQUE produces identical results irrespective of the order in which the input records are presented. In addition, it generates cluster descriptions in the form of DNF expressions [AGGR98] for ease of comprehension. Moreover, empirical evaluation shows that CLIQUE scales linearly with the number of instances, and has good scalability as the number of attributes is increased.

The algorithm WaveCluster [SC+98] works with numerical attributes and has an advanced multi-resolution. The main idea is to transform the original feature by applying wavelet transform and then find the dense regions in the new space. A wavelet transform is a signal processing technique that decomposes a signal into different frequency sub bands. The first step of the WaveCluster algorithm is to quantize the feature space. In the second step, discrete wavelet transform is applied on the quantized feature space and hence new units are generated. WaveCluster connects the components in 2 set of units and they are considered as cluster. Corresponding to each resolution of wavelet transform there would be set of clusters k, where usually at the coarser resolutions number of cluster is less. In the next step, WaveCluster labels the units in the feature space that are included in the cluster. WaveCluster gives high quality of clusters, can work well in relatively high dimensional spatial data and can successfully handle outliers. The algorithm's complexity is $O(N)$ for low dimensions, but with the increase in the number of dimensions it grows exponentially. Unlike other clustering methods, WaveCluster [SC+98] does not require

users to give the number of clusters. It is a very powerful method and automatically removes outliers, however, it is not efficient in high dimensional space.

2.2.5 Model based

AutoClass [CS96] uses the Bayesian approach, starting from a random initialization of the parameters, incrementally adjusts them in an attempt to find their maximum likelihood estimates. Another model based method is the SOM net [Koh95] which is based on a single layered neural network. The data objects are organized with a simple 2-D grid structure. Each neuron of the neural network is associated with a reference vector, and each data point is mapped to the neuron with the closest reference vector. In the process of running the algorithm, each data object acts as a training sample which directs the movement of the reference vectors towards the denser areas of the input vector space, so that those reference vectors are trained to fit the distributions of the input dataset. When the training is complete, clusters are identified by mapping all data points to the output neurons. An important property of the SOM is that it is very robust. The outlier can be easily detected from the map, since its distance in the input space from other units is large. SOM can deal with missing data values, too. It generates intuitive cluster patterns of a high dimensional dataset. However, it suffers from some disadvantages such as the number of clusters and the grid structure of the neuron map need to be given as input. It is also sensitive to the input parameter.

Though the model based approach can be considered as more relevant to the data mining problem, most of the existing methods under the approach suffer from the following disadvantages: (i) they try to fit a mathematical model to the data which may not be effective to all domains, (ii) the number of clusters and the grid structure need to be given as input, (iii) they are sensitive to the input parameters and (iv) the algorithms are not cost effective.

2.2.6 Graph Based

Graph theoretical clustering techniques represent the data in terms of a graph, thus converting the problem of clustering a dataset into such graph theoretical problems as finding minimum cut or maximum cliques in a proximity graph [BDSY99]. AUTOCLUST [LEC00] is a graph based algorithm that automatically extracts boundaries based on Voronoi modeling and Delaunay Diagrams. Parameters required are not specified by users but are revealed from the proximity structures of the Voronoi modeling, and AUTOCLUST calculates them from the Delaunay Diagram. This removes human-generated bias and also reduces the exploration time. The advantages are: (i) it is effective in the detection of clusters of different densities, and (ii) it identifies and removes multiple bridges linking clusters and has a complexity of $O(NlogN)$. CLuster Identification via Connectivity Kernels (CLICK) [SS00] tries to identify clusters as a highly connected component in a proximity graph based on a probabilistic assumption and can detect intersecting clusters. Cluster Affinity Search Technique (CAST) [BDSY99] is based on the concept of a corrupt clique graph data model. CAST assumes that the true clusters of the data points are obtained by a disjoint union of complete sub-graphs where each clique represents a cluster; where a cluster is a set of high affinity elements subject to a threshold. CAST discovers clusters one at a time. Both CAST and CLIQUE are popular for detecting clusters over gene expression data. The graph theoretic approach can be considered to be more relevant to gene expression data mining as they are capable of discovering intersected and embedded clusters. However, it sometimes generates non-realistic cluster patterns.

There are many applications that require the clustering of large amounts of high dimensional data. However, most automated clustering techniques do not work effectively and/or efficiently on high dimensional data, i.e. they often miss clusters with certain unexpected characteristics. The reasons for this are: (i) it is difficult to estimate the necessary parameters for tuning the clustering algorithms to the specific application's characteristics, (ii) it is hard to verify and interpret the resulting high dimensional clusters and (iii) often the concept of clusters inspired from low dimensional cases cannot be extended to high dimensional cases. A solution to these problems may be obtained by integrating all the requirements into a single algorithm

and to try to build a combination of clustering algorithms (ensembles of clustering algorithms)

2.2.7 Ensembles of Clustering Algorithms

In the combination of techniques in a group or ensemble, the outputs provided by different techniques are combined by one of several strategies in order to provide a consensus output value [HCFdC09]. The main goal is to use the best features of each individual technique and improve the overall performance in terms of accuracy or precision. The theoretical foundation of combining multiple clustering algorithms is still in its early stages. According to [HK07, HKK05], clustering ensembles are formed by the combination of a set of partitions previously produced by several runs of a single algorithm or by a set of algorithms. Since, there is no label associated with each object, some form of sophisticated strategies are needed in order to combine partitions found by different algorithms or different runs of the same algorithm in a consensus partition. Combining multiple clustering algorithms is a more challenging problem than combining multiple classifiers. Clustering combination a difficult task because various clustering algorithms produce very different results due to different clustering criteria, combining these clustering results directly may not generate a good meaningful result. According to [SG03], cluster ensembles can be formed in a number of different ways, such as (i) the use of a number of different clustering techniques (either deliberately or arbitrarily selected), (ii) the use of a single technique many times with different initial conditions and/or (iii) the use of different partial subsets of features or patterns. In [FJ02], a split-and-merge strategy is followed. In the first step, k-means algorithm is used to generate small, compact clusters. An ensemble of clustering algorithms is produced by random initializations of cluster centroids. Data partitions present in these clustering are mapped into a new similarity matrix between patterns, based on a voting mechanism. This matrix, is independent of data sparseness, is then used to extract the natural clusters using the single link algorithm. In [AK03], multiple clustering algorithms were combined based on a Weighted Shared nearest neighbors Graph method. In [YAL+06] multiple crossover repetitions were used to combine partitions created by different clustering algorithms. Each pair selected for a crossover operation should present a high overlap

in the clusters. The initial population comprises of all clusters created by the clustering algorithms used in the ensemble. This method, named heterogeneous clustering ensemble (HCE), differ from other ensemble approaches by taking characteristics from the individual algorithms and the dataset into account during the ensemble procedure. This method was compared with individual clustering algorithms using a gene expression dataset.

Due to the increasing size of current databases, constructing efficient distributed clustering algorithms has attracted considerable attention.

2.2.8 Distributed Clustering

Distributed Clustering assumes that the objects to be clustered reside on different sites. Instead of transmitting all objects to a central site (also known as server) where we can apply standard clustering algorithms to analyze the data (also known as sequential clustering), the data are clustered independently on different local sites. The central site updates the global clustering based on the local models, i.e. the representative clustering transmitted from the local sites. Generally, as far as distributed clustering is concerned, there are different scenarios: (i) Feature-Distributed Clustering (FDC), combines a set of clusterings obtained from clustering algorithm having partial view of the data features, (ii) Object-Distributed Clustering (ODC), combines clusterings obtained from clustering algorithm that have access to the whole set of data features and to a limited number of objects, and (iii) Feature/Object-Distributed Clustering (FODC), consists in combining clusterings obtained from clustering algorithm having access to limited number of objects and/or features of the data. Various distributed clustering techniques have been proposed such as a parallel version of the k-means algorithm was proposed in [DM99], a parallel version of DBSCAN, called PDBSCAN was presented in [XJK99] that uses a shared-nothing architecture with multiple computers interconnected through a network. PDBSCAN offers nearly linear speedup and has excellent scale-up and size-up behavior. The Density Based Distributed Clustering(DBDC) algorithm [JKP03] can be used in the case when the data to be clustered is distributed and infeasible to centralize. A detailed survey of distributed clustering is reported in Section 5.2.

2.2.9 Soft Computing

Traditional clustering approaches generate disjoint groups or clusters. Fuzzy clustering on the other hand associates each pattern with every cluster using a membership function with larger membership values indicating higher confidence in the assignment of the pattern to the cluster. One widely used fuzzy clustering algorithm is the Fuzzy C-Means (FCM) algorithm [Bez81a], which is based on k-means. FCM attempts to find the most characteristic point in each cluster, which can be considered as the cluster center and, then, the degree of membership for each object in the clusters are computed. The work in [AZM06] attempts to segment satellite image based on FCM algorithm and to detect different road classes on it. Some variants of fuzzy clustering for satellite image domain are presented in [ACN09, AN09, GyFSlXr09]. In [VB09], a density based clustering method called rough-DBSCAN is presented. It is a modication of the well known density based clustering method DBSCAN [EKSX96] and aims at achieving similar result as DBSCAN but in much smaller time requirement ($O(N)$).

Neural Networks-based clustering approaches have also gained popularity in recent years. Examples are SOFM (Self Organizing Feature Map) [Koh95, AN09] and ART (Adaptive Resonance Theory) [THHK02]. SOFM attempts to visualize a high dimensional input pattern with prototype vectors in a two-dimensional lattice structure, where each node in the lattice structure is a neuron, which are connected to each other via adaptable weights. During the training process, the neighboring input patterns are projected into the lattice corresponding to adjacent neurons. The advantages of SOFM are: (i) It enjoys the benefits of input space density approximation and, (ii) it is input order independent. The disadvantages are (i) like k-means, SOFM needs to predefine the size of the lattice, (the number of clusters) and, (ii) it may suffer from input space density misrepresentation. ART [THHK02] is a large family of neural network architecture and is capable of learning any input pattern in a fast, stable and self-organizing way.

Genetic Algorithms (GA) [Gol89] are also used in cluster analysis. GA clustering is basically a randomized search and optimization technique based on the principles of

evolution and natural genetics. Several GA-based clustering algorithms are found in the literature [ACN09, LLF+04b, KM99, LLF+04a, BP01, MB03a, BMM07a]. GAs have also been used to cluster satellite images such as the real-coded variable string length genetic fuzzy clustering in [MB03a] and multi-objective optimization algorithm in [BMM07a]. GGA (Genetically Guided Algorithm) is a genetic algorithm for fuzzy and hard k-means [HOB99]. Evolutionary techniques rely on certain parameters to empirically fit data and have high computational costs that limit their application in data mining. However, usage of combined strategies (e.g., generation of initial guess for k-means) has been attempted [BM93].

Other soft clustering algorithms have been developed and most of them are based on the Expectation-Maximization (EM) algorithm [DLR77]. They assume an underlying probability model with parameters that describe the probability that an instance belongs to a certain cluster. EM algorithm starts with initial guesses for the mixture model parameters. These values are then used to calculate the cluster probabilities for each object which in turn are used to re-estimate the parameters, and the process is repeated. A drawback of such algorithms is that they tend to be computationally expensive. Another problem found in this approach is called overfitting. This problem might be caused due to two reasons. On one hand, a large number of clusters may be specified and on the other, the distributions of probabilities have too many parameters. In this context, one possible solution is to adopt a fully Bayesian approach, in which every parameter has a prior probability distribution.

In case of fuzzy clustering, the problem of specifying the number of clusters exists. The basic advantage of ART is that it is fast, exhibits stable learning and pattern detection. The disadvantage is its inefficiency in handling noise and higher dimensional representation for clusters. GA-based clustering has also been used extensively recently, however, solutions are not always free from the local optima problem.

2.2.10 Subspace Clustering

Subspace clustering was initially proposed by Agrawal et al. [AGGR98], to evaluate features on only a subset of the data, based on a measure referred to as a "measure

of locality" representing a cluster. In this subsection, subspace clustering algorithms are discussed in four broad categories.

a. Bottom-Up *SubspaceSearch* Methods take the advantage of downward closure of the property of density to reduce the search space. It determines locality by creating bins for each dimension which finally form multi-dimensional grid, achieved by two approaches: (i) static grid-sized approach, e.g., CLIQUE [AGGR98] and ENCLUS[CFZ99], two popular algorithms of this category, and (ii) data driven strategies adopted to determine the cut-points, e.g., MAFIA, CBF, CL Tree and DOC. CLIQUE [AGGR98] can find clusters within subspaces using a grid-density based clustering approach and is capable of identifying arbitrary shaped clusters in any number of dimensions without specifying the number of clusters. The clusters may be found in the same space or in overlapping and disjoint subspaces. CLIQUE scales well with the number of instances and dimensions in the dataset. This method is not without its disadvantages, which are both grid size and the density threshold are input parameters which affect the quality of the clustering results, the small but important clusters can sometimes be eliminated during the pruning stage. ENCLUS [CFZ99]is a bottom-up clustering method which defines clusters based on entropy and can locate overlapping clusters of various shapes in subspaces of different sizes. However, its scalability is poor with respect to the subspace dimensions. Merging of Adaptive Finite Intervals Algorithm (MAFIA) [GGN+99] is a variant of CLIQUE and uses an adaptive, grid-based approach with parallelism, to improve scalability. The advantages of MAFIA are that it can locate clusters of various sizes and shapes, its performance is faster than CLIQUE due to the adoption of parallel approach and its scale-up is linear. However, the running time grows exponentially as the number of dimensions increase. Cell-Based Clustering (CBF) [CJ02] is also a bottom-up algorithm but unlike CLIQUE and MAFIA, it uses an efficient algorithm for creation of partitions optimally, to avoid exponential growth of bins with the increase in the number of dimensions. CBF locates clusters of various sizes and shapes, scales linearly with respect to the number of records in a dataset and its performance is better since the bins are stored in an index structure. But, it is sensitive

to the threshold which determines the bin frequency of a dimension and the threshold which determines the number of data points in a bin. Density-based Optimal projective Clustering (DOC) [PJAM02] is basically a hybridization of bottom-up and top-down approaches. It introduces the notion of an optimal projective cluster. The advantage is that the running time grows linearly with the number of instances, whereas the disadvantages are it is sensitive to the input parameters, is able to identify mostly hyper-rectangular shaped clusters, and the running time grows exponentially with the increase in the number of dimensions in the dataset.

b. Top-Down *SubspaceSearch* Methods start with an initial approximation of clusters over an equally weighted full feature space. Next, it follows an iterative procedure to update the weights and accordingly reforms the clusters. It is an expensive clustering algorithm over the full feature space. However, the use of sampling technique can improve the performance. The number of clusters and the size of the subspace are the most critical factors in this approach. PROCLUS [AWY+99] is a sampling biased top-down subspace clustering algorithm which randomly selects a set of k-medoids from a sample and iteratively improves the choice of medoids to form better clusters. The disadvantages of this method are it is biased towards hyper-spherical shaped clusters, clustering quality depend upon the size of the sample chosen, and it is sensitive to the input parameters. ORCLUS [AY00] attempts to form clusters iteratively by assigning the points to the nearest cluster representation. It computes the dissimilarity between a pair of points as a set of orthonormal vectors over a subspace. It is a fast and a scalable method. However, it requires the size of the subspace dimensionality and the number of clusters apriori and may sometimes miss some small clusters. The algorithm δ-Clusters [YWWY02] starts with an initial seed and attempts to improve the overall quality of the cluster iteratively by swapping dimensions with instances. The advantage is that the use of coherence as a similarity measure makes it more relevant for microarray data analysis and its disadvantages are that (i) it is dependent on two input parameters which are number and size of the cluster (ii) the running time is dependent upon the cluster size. COSA [FM04] starts with an equally weighted

dimension for each instance and then it examines the k-nearest neighbors (knn) of an instance. Based on knn, it calculates the respective dimension weights for each instance and assigns higher weighted dimensions to those instances which have lesser dispersion within the neighborhood. This process is then repeated with the new instances till the weights stabilize. The advantage of this method is that the number of dimensions in clusters need not be specified and the dimension weights are calculated.

c. *Biclustering* Algorithms: A bicluster [CC00] is an $I \times J$ sub-matrix that exhibits some coherent tendency where I and J are set of genes (rows) and conditions (columns), respectively, and $|I| \leq |G|$ and $|J| \leq |T|$. The volume of a bicluster (I, J) is defined as the number of elements e_{ij} such that $i \in I$ and $j \in J$. The quality of a bicluster is assessed based on the mean squared residue, which is the variance of the set of all elements in the bicluster, plus the mean row variance and the mean column variance. The lower the mean squared residue, stronger is the coherence exhibited by the cluster and better is the quality of the bicluster. Cheng and Church [CC00] were the pioneers in applying biclustering to gene expression data. To obtain the larger bi-clusters with minimum mean squared residue, the authors introduced the node addition method which simultaneously adds rows/columns as deletions took place. However, it is not capable of identifying overlapping/embedded clusters because the elements of the already identified bicluster are masked by random noise. FLexible Overlapped biClustering (FLOC) [YWWY03] addresses the limitation of Cheng and Church [CC00] and accelerates the biclustering process, FLOC uses a probabilistic algorithm which can discover a set of k-possible overlapping biclusters simultaneously. Order-preserving submatrices (OPSM) [BdCKY02] is another probabilistic model which attempts to address the idea of large OPSMs with maximum statistical significance. Tanay et al. in [TSS02] introduced Statistical-Algorithmic Method for Bicluster Analysis (SAMBA), a bi-clustering algorithm that performs simultaneous bicluster identification by using exhaustive enumeration. An important advantage of SAMBA is that it is capable of analyzing large datasets in lesser time. The Coupled Two-Way Clustering (CTWC) [GLD00] tries to identify couples of small subsets of fea-

tures and objects. The Interrelated Two-Way Clustering (ITWC) [TZRZ01] is an iterative biclustering algorithm based on a combination of the results obtained by clustering performed on each of the two dimensions of the data matrix separately.

d. *TriClustering* Algorithms tries to find coherent clusters along gene-sample-time (temporal) or gene-sample-region (spatial) dimensions, known as triclusters, which may be arbitrarily positioned and overlapped [ZZ05]. TriClustering algorithms are used for mining such coherent clusters in three-dimensional gene expression datasets. TriCluster relies on a graph-based approach to mine various types of clusters, including clusters having constant or similar values along each dimension with scaling and shifting expression patterns, based on different parameter values.

2.2.11 A General Comparison among Different Approaches

We have discussed some of the unsupervised clustering methods present in the literature. Partitioning algorithms typically represent clusters by a prototype and an iterative control strategy is used to optimize the whole clustering such as the average or squared distances of instances to its cluster centers (prototypes) are minimized. Partitional clustering algorithms are effective for determining clusters of convex shape, similar size and density, and if the number of clusters can be reasonably estimated. However, determining the appropriate number of clusters is very difficult. Hierarchical clustering algorithms decompose the dataset into several levels of partitioning and are represented by a tree structure which splits the dataset recursively into smaller subsets. Although hierarchical clustering algorithms can be very effective in knowledge discovery, the cost of creating the tree is very expensive for large datasets. In density-based approaches clusters are regarded as regions in the data space where the objects are dense, and they are separated by regions of low density (noise). These regions may have an arbitrary shape and the objects inside a region may be arbitrarily distributed. Generally, grid-based clustering algorithms first separate the clustering space into a finite number of cells (hyper-rectangles) and then perform the required operations on the quantized space. Cells that contain more than certain number of

points are treated as dense and the dense cells are connected to form the clusters. A solution for better results could be instead of integrating all the requirements into a single algorithm, to try to build a combination of clustering algorithms. In addition, the impact of various soft computing techniques is also considerable for identifying clusters that are not crisp. The performance and quality of distributed and parallel clustering techniques have helped in managing and processing massive data. For high dimensional clustering, subspace clustering algorithms have given quite good results. A performance comparison of various clustering algorithms is given in Table 2.1.

2.2.12 Handling Outliers

Data usually have an associated amount of noise, which can be viewed as outliers. Alternately, outliers can be viewed as legitimate records having abnormal behavior. The algorithm BIRCH [ZRL96] revisits outliers during the major CF tree rebuilds, but in general handles them separately. Some algorithms have specific features for outliers handling. The algorithm CURE [GRS98] uses shrinkage of cluster representatives to suppress the effects of outliers. K-medoids methods are generally more robust than k-means methods with respect to outliers. The algorithm DBSCAN [EKSX96] uses concepts of internal (core), boundary (reachable), and outliers (non-reachable) points. CLIQUE [AGGR98] eliminates subspaces with low coverage. WaveCluster [SC+98] handleS outliers very well through its filtering process. The algorithm ORCLUS [AY00] produces a partition plus a set of outliers. A point is said to be an outlier if its ε-neighborhood contains less than $MinPts$-fraction of a whole dataset D [KNT00]. In essence, different subsets of data have different densities and may be governed by different distributions. A point close to a tight cluster can be a more probable outlier than a point that is further away from a more dispersed cluster. The concept of local outlier factor (LOF) that specifies a degree of outlier-ness comes to rescue [BKNS00]. The definition is based on the distance to the k-nearest neighbor. Knorr et al. [KNZ01] addressed a related problem of how to eliminate outliers in order to compute an appropriate covariance matrix that describes a given locality. Outlier detection techniques can be divided into the following categories given below.

- *Distribution-based methods* handle one dimensional data and assume a statistical distribution such as Gaussian and try to fit the data to the model by estimating the parameters such as mean and variance from the data [BL94]. They vary in terms of type of distribution, number of outliers to be identied and type of outliers. Then they employ a test based on the distribution property to identify outliers w.r.t. this distribution. In reality, prior knowledge about the distribution of the dataset is not always available.

- *Depth-based approaches* [RR96, JKN98] employ computational geometry to compute different layers of convex hulls and declare those objects in the outer layer as outliers. However, they suffer from the dimensionality curse and cannot cope with large dimensions.

- *Distance-based approaches* distinguish points which are likely to be outliers from others based on the number of points in their neighborhood and are suitable for nding outliers in large datasets. Corresponding to clustering algorithms that nd convex clusters [KR90, NH02], one well known technique is the $DB(p, d)$-outlier [KN98], where a point in a dataset D is an outlier if at least p fraction of points in D lie greater than distance d from it. A special case of $DB(p, d)$-outlier is proposed in [RRS00], where the distance to the k-th nearest neighbor is used to rank the outlyingness. But, it cannot handle data with different local densities and hence can only find global outliers. Besides, the users parameters, such as p, d, k, are hard to determine.

- *Density-based approaches* focus on the local density comparison only with the immediate neighbors. They come in two classes, subspace and full space. Sometimes, an object could reside in a low density region only in a subspace, which is obtained by projecting the original full space onto one of its subsets. In [AGGR98], all possible subspaces were searched where there are regions with much lower density than the rest of the subspace. All points in those low density regions are declared as outliers. In [BKNS00], the authors introduced the notion of LOF, which measures the degree of outlyingness, based on the difference in the local density of a point and its k-nearest neighbors. $DB(p, d)$-outlier cannot detect local outliers w.r.t. a neighboring dense cluster in presence of

another very sparse cluster because although the local density of the outlier can be lower than those inside the neighboring high density cluster, it may be comparable to those inside the sparse (low density) cluster, therefore, a large portion of points in the sparse cluster will also be classied as outliers. LOF solves this problem by comparing local density of the outlier only with those of its neighboring objects. The weakness of LOF is that it cannot detect outliers whose local density is higher, not lower, than those inside the neighboring pattern. Such a pattern may consist of a set of regularly spaced points that have lower densities than their neighboring outliers.

2.3 Discussion

This chapter presents various proximity measures, clustering and outlier detection techniques. Clustering algorithms are dependent on the proximity measure chosen. Moreover, there exists no particular measure which can handle all the issues or domains. From the discussion above, we conclude that various clustering algorithms require different types of input parameters and clustering results are highly dependent on the values of the parameters. Clustering algorithms that can handle massive data, identify clusters even in high dimensional and noisy data are in great demand. Again, identification of variable density clusters is an important research area which has gained focus due to its huge potential. Clustering algorithms that do not require the number of clusters beforehand, insensitive to the proximity measure, and robust to noise are of utmost importance.

In this thesis, we present several clustering techniques for application over 2D spatial data, satellite and gene expression data. The advantages of our techniques are: (i) Independence of the number of clusters, (ii) detection of variable density clusters, (iii) identifications of sub-clusters, (iv) capability in detecting clusters in large-scale data and (v) handling outliers and noise. The next chapter presents our first clustering technique capable of identifying variable density clusters in 2D spatial data.

Table 2.1: Comparison of various clustering algorithms

Algorithms	*No. of Parameters*	Optimized For	Structure	*Multi-Density Clusters*	*Embedded Clusters*	*Complexity*	*Noise Handling*
k-means	No. of clusters	Separated Clusters	Spherical	No	No	$O(l^tkN)$	No
k-modes	No. of clusters	Separated Clusters, large datasets	spherical	No	No	$O(l^tkN)$	No
FCM	No. of clusters	Separated Clusters	Non-convex shapes	No	No	$O(N)$	No
PAM	No of clusters	Separated Clusters, small datasets	spherical	No	No	$O(l^tk(N-k)^2)$	No
CLARA	No. of clusters	relatively large datasets	spherical	No	No	$O(k\times Sz^2+k(N-k))$	No
CLARANS	No. of clusters, Max no. of neighbors	better than PAM, CLARA	spherical	No	No	$O(kN^2)$	No
BIRCH	Branching factor, Diameter, Threshold	Large data	Spherical	No	No	$O(N)$	Yes
CURE	No. of clusters, No. of representatives	Any shape large data	Arbitrary	No	No	$O(N^2logN)$	Yes
ROCK	No. of clusters	Small noisy data	Arbitrary	No	No	$O(N^2+Nm_mm_a+N^2logN)$	Yes
CHAMELEON	3 (k-nearest neighbor, MINSIZE, α^c)	Small datasets	Arbitrary	Yes	No	$O(N^2)$	Yes

Algorithms	*No. of Parameters*	Optimized For	Structure	*Multi-Density Clusters*	*Embedded Clusters*	*Complexity*	*Noise Handling*
DBSCAN	2 ($MinPts$, ε)	Large datasets	Arbitrary	No	No	$O(NlogN)$ using R^* tree	Yes
OPTICS	3 ($MinPts$, ε, ε')	Large datasets	Arbitrary	Yes	Yes	$O(NlogN)$ using R^* tree	Yes
DENCLUE	2 ($MinPts$, ε)	Large datasets	Arbitrary	No	No	$O(NlogN)$ using R^* tree	Yes
WaveCluster	No. of cells for each dimension, No of applications of transform	Any shape, Large data	Any	Yes	No	$O(N)$	Yes
STING	No. of cells in lowest level, No. of objects in cell	Large spatial datasets	Vertical and horizontal boundary	No	No	$O(N)$	Yes
CLIQUE	Size of the grid, minimum no. of points in each grid cell	High dimensional, large datasets	Arbitrary	No	No	$O(N)$	Yes
MAFIA	Size of the grid, minimum no. of points in each grid cell	High dimensional, large datasets	Arbitrary	No	No	$O(c^{k'})$	Yes
AUTOCLUST	NIL	Massive data	Arbitrary	No	No	$O(NlogN)$	Yes

Chapter 3

Grid-Density based Spatial Data Clustering

Spatial information, also known as geo-spatial or geographic information identify geographic locations of features and boundaries on earth. Among data mining techniques, clustering has been widely used to mine information from such data. Spatial data may contain data of different densities and the identification of clusters in variable density regions is an important research issue.

This chapter presents a grid-density based clustering technique that can identify embedded clusters in variable density space. The proposed technique can detect clusters for different density regions and can also detect embedded clusters. The technique has also been slightly enhanced for outlier detection. Experimental results using several test and standard synthetic datasets establish the superiority of this technique in terms of cluster quality.

3.1 Introduction

Spatial data contains information about the geographic location of features and boundaries on earth, such as natural or constructed features and oceans. Spatial information is usually stored in terms of coordinates and topology, and can be mapped for display. Spatial data is also stored in terms of pixels. From such data, information units such as points, lines, regions, partitions (maps) and graphs (networks) can be extracted. Spatial data is often accessed, manipulated or analyzed through Geographic Information Systems (GIS). Spatial data may also contain non-spatial attributes that may be either dependent or independent of location. Spatial data is a specialized domain for data mining. Its aim is to discover latent or implicit knowledge in spatial data. Spatial data mining has the end objective of finding patterns in geography. The immense explosion in geographically relevant data created by developments in IT, digital mapping, remote sensing, and the global diffusion of GIS emphasizes the importance of developing data driven inductive approaches to geographical analysis and modeling. Data mining, which is the automated search for hidden patterns in large databases, offers great potential benefits for applied GIS-based decision-making. Geo-spatial data repositories tend to be very large. Spatial cluster analysis plays an important role in quantifying geographic variation patterns. It is commonly used in disease surveillance, spatial epidemiology, population genetics, landscape ecology, crime analysis and many other fields. The clustering task becomes more challenging as the number of embedded patterns increases with the increase in variable density regions. This chapter provides an overview of spatial cluster analysis. Initially, we survey different forms of clustering applicable to spatial data. Then, we discuss a new clustering technique for spatial data.

3.2 Related Work

Clustering techniques have been classified into partitional, hierarchical, density based, grid based and model based approaches. Among these techniques, the density-based approach stands out for its ability to discover arbitrary shaped clusters of good quality even in noisy datasets [EKSX96]. The grid-based clustering approach is well

known for its fast processing time especially for large datasets. This section provides the background research on density based and grid based clustering.

3.2.1 Density based approach

The idea behind density based clustering approach is that the density of objects within a cluster is higher compared to the objects outside of it. DBSCAN [EKSX96] is a density-based clustering algorithm capable of discovering clusters of various shapes even in presence of noise. The key idea of DBSCAN is that for each object of a cluster, the neighborhood of a given radius (ε) must contain at least a minimum number of objects ($MinPts$) and the density in the neighborhood must exceed some threshold. Thus it has two parameters ε and $MinPts$. It is difficult for users to correctly predict suitable values of these two parameters for a dataset. Another drawback of this technique is the high computational complexity because it examines all the neighborhoods to check if the core condition is satisfied for each object. This step is very expensive especially when the algorithm runs on very large datasets. Therefore, spatial index structures are used for large datasets. For massive datasets, it becomes very time consuming, even if we use efficient data structures such as R* tree. An additional drawback of DBSCAN is that due to the use of the global parameters, it fails to detect embedded or nested clusters. OPTICS [ABKS99] can identify embedded clusters in space of varying densities. However, its execution time is high for large datasets with spaces of variable densities and nested cluster structures. Moreover, OPTICS is highly sensitive to its parameters $MinPts$, ε and ε' (core distance).

3.2.2 Grid based approach

Grid based methods divide the data space into a finite number of cells that form a grid structure on which the clustering is performed. There is a high probability that all data objects that fall into the same grid cell belong to the same cluster. Therefore all data objects belonging to the same cell can be aggregated and treated as one object [HC04]. It is due to this reason that grid-based clustering algorithms are computationally efficient. Such algorithms have many advantages such as the total

number of the grid cells is independent of the number of data objects and is insensitive to the order of input data objects. Some popular grid based clustering techniques are discussed in [WYM97, NGC00]. STING [WYM97] uses a multi-resolution approach to perform cluster analysis. The advantage of STING is that it is query-independent and easy to parallelize. However the shapes of clusters are restricted to horizontal or vertical boundaries but no diagonal boundary is possible. WaveCluster [SC+98] also uses a multidimensional grid structure and detects clusters of data at varying levels of accuracy. It automatically removes outliers and is very fast. However, it is not suitable for high dimensional datasets. CLIQUE [AGGR98] is a hybrid clustering method that combines both density-based and grid-based approaches. It automatically finds subspaces of the highest dimensionality and is insensitive to the order of input. Moreover, it scales well as the number of dimensions in the data increases. However, the accuracy of the clusters may degrade because the method is simple. pMAFIA [NGC00] is an optimized and improved version of CLIQUE. It detects coarse clusters and scales exponentially as number of dimensions in the dataset increases. GDILC [YJ01] is based on contour figures, density-based and grid-based clustering algorithms. This algorithm needs expensive neighborhood distance calculation with each of the cell's neighbors. Therefore, the cost of GDILC is too high, especially for large datasets.

3.2.3 Clustering in multi-density and variable density data space

One major application of clustering spatial databases is to find clusters of spatial objects which are close to each other. Most traditional clustering algorithms discover clusters of arbitrary densities, shapes and sizes. Very few clustering algorithms show desirable efficiency when clustering multi-density datasets. This is also because small clusters with small number of objects in a local area are easily missed by a global density threshold. Some clustering algorithms that can cluster multi-density datasets are Chameleon [KHK99], SNN [ESK03] (shared nearest neighbor) and the multi-stage density-isoline algorithm [YcMFJd03]. Chameleon [KHK99] can handle multi-density datasets, but for large datasets the time complexity is too high. SNN

[ESK03] can find clusters of varying shapes, sizes and densities. The disadvantage of SNN is that the degree of precision is low in identifying multi-density clusters and in handling outliers. The multi-stage density-isoline algorithm [YcMFJd03] clusters datasets in multiple stages. The disadvantage of the algorithm is that clusters cannot be separated efficiently. DGCL [KGX+06] is based on a grid-density based clustering approach. However, due to the use of a uniform density threshold, the low density clusters are often lost.

Most real life datasets have a skewed density distribution and may also contain nested cluster structures, the discovery of which is very difficult. OPTICS [ABKS99] handles datasets with variable density successfully. OPTICS has been discussed in detail in Section 2.2.3 of this thesis. A density based clustering technique, LDBSCAN, that can detect both embedded as well as overlapping clusters using local density information is presented in [DXLG06]. EnDBSCAN [RB05] makes an attempt to detect embedded or intrinsic clusters using an integrated approach. Based on our experimental analysis using very large synthetic datasets, we conclude that EnDBSCAN can detect embedded clusters. However, with the increase in the volume of data, its performance degrades. EnDBSCAN is highly sensitive to the parameters $MinPts$ and ε. In addition to these two parameters, OPTICS requires an additional parameter $\varepsilon\prime$. DD_DBSCAN [BB07] finds clusters of different shapes and sizes that differ in local densities. However, the method is unable to handle the density variation within a cluster, i.e. a cluster may have wide density variation from one end to another. A Density Differentiated Spatial Clustering Technique (DDSC), proposed in [BB08], is also an extension of the DBSCAN method. It detects clusters that have non-overlapped spatial regions with reasonable homogeneous density variations within them. If there is significant change in densities of adjacent regions, then such regions are separated into different clusters. An added advantage is that the sensitivity to the input parameter ε, which is an important disadvantage of DBSCAN, is reduced significantly. EDBSCAN [AAS+09] is another improvement of DBSCAN. It keeps track of density variation within a single cluster. It calculates the density variance of a core object with respect to its ε-neighborhood. If density variance of a core object is less than or equal to a threshold value and also satisfies

the homogeneity index in its ε-neighborhood, it allows the core object to expand. It calculates the density variance and homogeneity index locally in the ε-neighborhood of a core object. Another method, VDBSCAN proposed in [LZW07], for analysis of varied-density datasets. VDBSCAN adapts the traditional DBSCAN algorithm using K-dist plots to select values of the parameter ε for different density regions. The DBSCAN algorithm is run for different values of ε to make sure that all the clusters of the corresponding density are discovered. In a run of DBSCAN, the objects that have already been clustered in previous runs are ignored; this avoids the problem of marking both denser areas and sparser ones as one single cluster later. Another method for detecting clusters of variable densities, DVBSCAN (Density Variation Based Spatial Clustering of Applications with Noise), is proposed in [RJJK10]. It starts the formation of the cluster by selecting a core object. Then it computes the Cluster Density Mean (CDM) of the growing cluster before allowing the expansion of an unprocessed core object. After that it computes the Cluster Density Variance (CDV) within the ε-neighborhood of the unprocessed core object with respect to Cluster Density Mean (CDM). If the Cluster Density Variance (CDV) of the growing cluster with respect to CDM is less than a specified threshold value, say, CDV_α and the difference between the minimum and maximum densities of objects in the ε-neighborhood of the objects of the growing cluster, including the ε-neighborhood objects of the unprocessed core object, is less than a specified threshold value CSI_λ then only an unprocessed core object is allowed for expansion otherwise the object is simply added into the cluster. This algorithm claims to handle variable density clusters in the dataset but is computationally expensive.

3.2.4 Discussion

Based on our survey, we observe that partitional methods detect only spherically shaped clusters, and sometimes cannot handle noise or detect outliers. Hierarchical algorithms give a tree view of the clusters but are dependent on a termination factor which is hard to derive apriori. Model based methods fit a mathematical model to the data but are computationally expensive and not all data fit into the model. It is also evident from our survey that Density Based methods are more efficient in finding clusters and in handling outliers/noises in large spatial datasets. However,

a basic disadvantage of these methods is their dependency on input parameters and the clusters detected are highly sensitive to these parameters. Hierarchical clustering algorithms fix the membership of a data object once it has been allocated to a cluster. In hierarchical clustering, our regular point-by-attribute data representation is sometimes of secondary importance. BIRCH [ZRL96], CURE [GRS98] and CHAMELEON [KHK99] use complex criteria for compressing and relocating data before merging clusters. After carefully considering the advantages of both density based and grid based approaches, this chapter presents a grid-density based clustering technique to enable (i) the detection of arbitrary shaped clusters, (ii) detection of variable density and embedded clusters, (iii) handling of outliers, and (iv) faster cluster discovery of finer clusters.

3.3 Grid-Density based Clustering Technique

The distribution of data in a dataset is not uniform in general. Some portions of the data space are highly dense while some portions are sparse. Therefore, the data space is divided into grid cells and the cells whose densities are similar are merged. These similarly dense grid cells are together form the coarse clusters or rough clusters. Once merging of grid cells according to density terminates, a set of coarse clusters is obtained. Thus, coarse clusters represent the maximal space that can be covered by the similar grid cells in terms of their densities. A method for computing the number of grid cells ($gr_n \times gr_n$) is given next.

Computing the Number of Grid Intervals (gr_n)

The following formula is used to estimate the number of grid intervals gr_n.

$$gr'_n = \sqrt{\frac{N}{M}} \tag{3.1}$$

$$gr_n = [gr'_n - 5, gr'_n + 5] \tag{3.2}$$

where N is the number of data objects and M is a positive integer to adjust the value of gr'_n. In fact, M stands for the average number of data samples in a cell. We carried out a large number of experiments to understand the relationship between

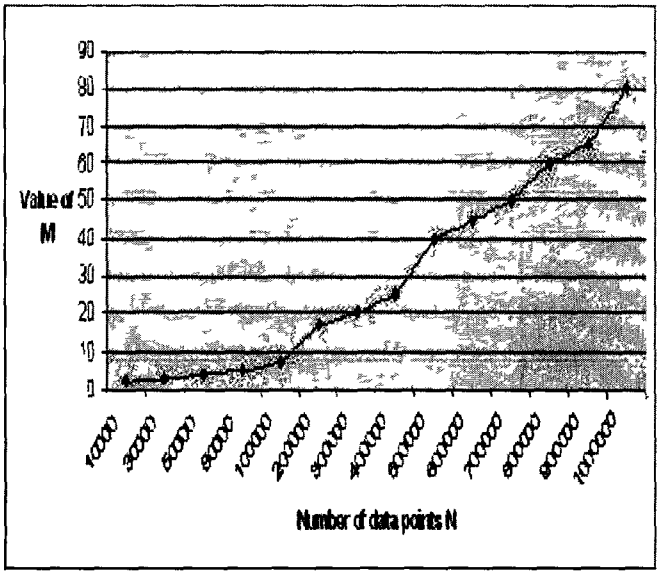


Figure 3.1: M depends on the number of data objects

the number of data objects (N) and values for M. A graph of N versus M is shown in Figure 3.1. The value of gr_n is calculated based on Equation 3.1 and Equation 3.2. Based on our wide range of experiments, we observe that gr_n varies within the range given in Equation 3.2.

Next, we introduce a few definitions to help in understanding the proposed algorithm.

3.3.1 Density based approach

Definition 1. Cell Density: It is defined as the number of objects within a grid cell.

Definition 2. Useful Cell: A cell is defined as a useful cell if it is populated, i e., it contains data objects within it

Definition 3. Neighbor Cell. A cell which is edge neighbor or vertex neighbor of a current cell is defined as the neighbor of the current cell Figure 3 2 (a) shows the neighbor cells (shaded) of the current cell P.

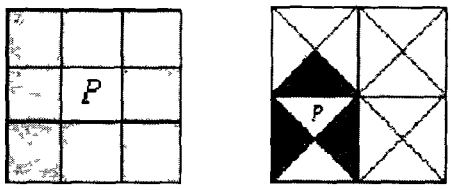


Figure 3.2: (a) The white cell is the current cell and all its neighbors are in gray (b) The white triangle P is the current triangle and all its neighbors are shaded gray.

Definition 4. Density Confidence of a cell: If the ratio of the densities of the current cell and one of its neighbors is less than some β' (given as user input), β' is the density confidence between them. Density confidence plays an important role in cluster formation. For two cells p_1 and q_1 to be merged into the same cluster, the condition $\beta' \leq d_n(p_1)/d_n(q_1)$ should be satisfied, where d_n represents the density of that particular cell.

Definition 5. Reachability of a cell: A cell p is reachable from a cell q if p is a neighbor cell of q and cell p satisfies the density confidence condition w.r.t. cell q.

The proposed grid-density based clustering technique incorporates a *Triangle-subdivision* based approach for better processing of the boundary objects to get finer clusters. A triangles is a degenerated quadrilaterals with two of the vertices merged together. We have found the *Triangle-subdivision* based approach for selection of the boundary objects to be more effective compared to the one based on rectangles. The following definitions are introduced to support the *Triangle-subdivision* based approach:

Definition 6. Triangle Density: The density of a triangle is defined as the number of objects within that particular triangle of a grid cell.

Definition 7. Useful Triangle: A triangle T_p in a cell p is defined as a useful triangle if it is populated with at least one object.

Definition 8. Neighbor Triangle: A triangle which has a common edge with the current triangle is said to be a neighbor of the current triangle. Figure 3.2 (b) shows the neighbor triangles (shaded) of the current triangle P.

Definition 9. Density Confidence of a triangle: If the ratio of the densities of the current triangle and one of its neighbors is less than $\beta'/4$, the two triangles can be merged into the same cluster. Therefore the following condition should be satisfied: $\beta'/4 \leq d_n(T_p)/d_n(T_q)$ where d_n represents the density of the particular triangle.

Definition 10. Reachability of a triangle: A triangle T_p is reachable from a triangle T_q if T_p is a neighbor triangle of T_q and triangle T_p satisfies the density confidence condition w.r.t. triangle T_q.

Definition 11. Cluster: A cluster is defined to be the set of objects belonging to a set of reachable cells and triangles. A cluster C_i w.r.t. β' is a non-empty subset satisfying the following condition,
$\forall p, q$: if $p \in C_i$ and q is reachable from p w.r.t. β', then $q \in C_i$, where p and q are either cells or triangles, respectively.

Both cell-reachability and triangle-reachable relation have symmetric and transitive properties within a cluster C_i.

Definition 12. Noise: Noise is simply the set of objects belonging to the cells (or triangles) not belonging to any of its clusters. Let $C_1, C_2, \cdots, C_k$ be the set of k clusters w.r.t. β'. Then

$$noise = \{no_p | p \in n \times n, \forall i : p \notin C_i\} \tag{3.3}$$

where no_p is the set of objects in cell p.

3.3.2 Density Confidence

The density confidence for a given set of cells reflects the general trend of that set. If the density of one cell is abnormal compared to that of the others it will not be included in the set. Similarly, each useful cell has a density confidence with each of its neighbor cells. If the density confidence of a current cell with one of its neighbor cells does not satisfy the density confidence condition, that neighbor cell is not included in the local dense area. On the contrary, if it satisfies the condition, we treat the neighbor cell as a part of the local dense area and merge the cell with the dense

area. In comparison to other methods of setting a global threshold, this method has the ability to recognize the local dense areas in the data space where multi-density clusters exist.

In light of the above definitions, following lemmas are stated.

Lemma 1. Let C_i be a cluster w.r.t. β' and let p be any cell in C_i. Also, let T_p be a triangle in p. Then C_i can be defined as the set of elements, $S = \{s \cup T_s | s$ is cell-reachable from p w.r.t. β' and T_s is triangle-reachable from T_p w.r.t. $\beta'\}$.

Proof. Suppose r is a cell or a triangle, where $r \in s \cup T_s$ and r is neither cell-reachable nor triangle-reachable from p w.r.t. β'. But, a cluster according to Definition 11 will be the set of objects which are cell-reachable or triangle-reachable from p. Therefore, we come to a contradiction and hence the proof. □

Lemma 2. A cell (or triangle) corresponding to noise objects is not cell-reachable (or tr$p_{i,j}$iangle-reachable) from any of the clusters. For a cell p we have, $\forall p : p$ is not reachable from any cell (or triangle) in C_i, i.e., $p \notin C_i$.

Proof. Suppose, C_i is a cluster w.r.t β' and let p be a cell (or triangle) corresponding to noise objects. Let p be cell-reachable (or triangle-reachable) from C_i. Then $p \in C_i$. But, this violates Definition 12 in that noise objects belong to cells that are neither cell-reachable nor triangle-reachable from any of the clusters. Therefore, we come to the conclusion that p is not reachable from any cell (or triangle) in C_i. □

Lemma 3. A cell (or a triangle) r can be cell-reachable (or a triangle-reachable) from only a single unique cluster.

Proof. Let C_i and C_j be two clusters w.r.t. β' and let p be any cell (or a triangle) in C_i and q be any cell (or a triangle) in C_j. Suppose a cell r is cell-reachable (or triangle-reachable) from both p and q. Then $r \in C_i$ and $r \in C_j$. This means that the clusters C_i and C_j should be merged. This violates the basic notion that clusters are unique sets. Thus, we can conclude that if r is cell-reachable (or a triangle-reachable) from p w.r.t. β', r is not cell-reachable (or a triangle-reachable) from q w.r.t. β', i.e. $r \in C_i$ and $r \notin C_j$. Therefore the lemma is proved. □

3.4 GDCT: A Grid-Density based Clustering using Triangle-subdivision

Initially, the GDCT algorithm divides the data space into $gr_n \times gr_n$ non-overlapping grid cells (where gr_n is a user input) and maps the dataset to each cell. It then calculates the density of each cell. The cells are sorted according to their density values. The result is an ordered sequence $< C_{p(i)} >$, where $p(i)$ denotes a permutation of the index, i, defining the sorted order of the cells. The algorithm uses the cell information (density) of the grid structure and clusters the data objects according to their surrounding cells. The cell with the highest density becomes the cluster *initiator*. The remaining cells are then clustered iteratively in order of their densities, thereby building new clusters or merging with existing clusters. Only the useful cells adjacent to a cluster can be merged. A neighbor search is conducted, starting at the highest density cell and inspecting adjacent cells. If a neighbor cell which satisfies the density confidence condition of a cell is found, then the neighbor cell is merged with the current cell to form the coarse cluster, and the search proceeds recursively with this neighbor cell. This search is similar to a graph traversal where the nodes represent the cells and an edge between two nodes exists if the respective cells are adjacent and satisfies the density confidence condition of a cell.

A coarse cluster is an approximation of the innermost cluster or the cluster with the maximum density, minus the boundary region. Cells falling inside a particular coarse cluster are classified with the same cluster_id. A coarse cluster is rough in its shape and size. Its shape in the boundary region of the cluster varies abruptly and rapidly since there is a transition from a denser region to sparser regions. Therefore, this region needs special analysis. So, after the coarse cluster is formed, there may still be some objects which are part of the cluster but have not been included in the coarse cluster as shown by the red ellipses in Figure 3.3. Since the objects inside these regions do not enter the coarse cluster though they should be part of the cluster, we expand the cells in the boundary region of the coarse cluster with the help of triangles. The objects inside the ellipses in the boundary region of the cluster have been left out because the cells in which they belong have not satisfied the density

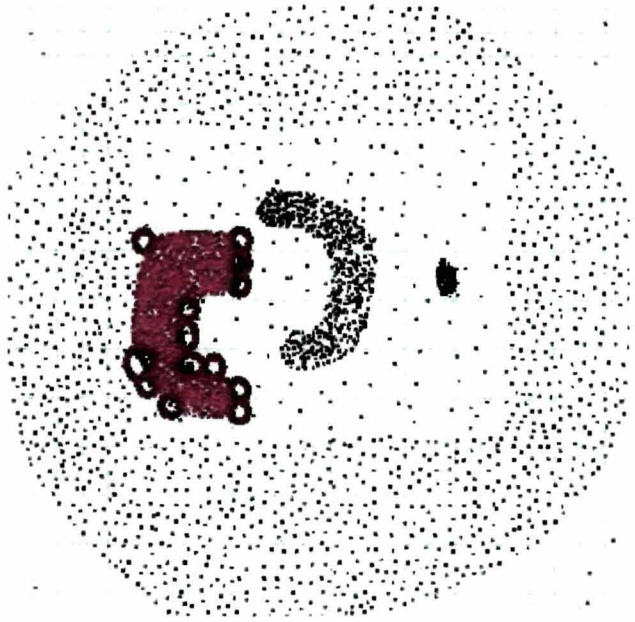


Figure 3.3: Example grid approximation for a dataset ($gr_n = 25$)

confidence. This is because only a small portion of that part of the cluster has fallen in a different cell and hence the density of that cell is much less than its neighbor which is in the coarse cluster.

Therefore, to find the finer cluster boundaries, the cells located in the boundary regions are triangulated by dividing into four triangles. Only those cells in the coarse cluster that have at least one of its useful neighbor cells as unclassified are triangulated. The cells which are unclassified and have at least one of its neighbor cells in the most recent coarse cluster formed are also triangulated. The objects of the cells that have been triangulated are mapped to the respective triangles in which they fall. Barycentric coordinates[1] are used to find which object falls in which triangle. This method was chosen since it is independent of the cyclic order of the vertices.

[1]Obtained from http//steve.hollasch.net/cgindex/math/barycentric.html

Procedure of GDCT

The algorithm includes the following steps:

1. Create grid structure.
2. Compute density of each cell.
3. Sort cells according to their densities.
4. Identify the maximum dense cell from the set of unclassified cells.
5. Traverse neighbor cells starting from the dense cell to form the coarse cluster.
6. Perform Triangle-subdivision in the coarse grid of the border cells that have at least a useful cell as one of its neighbors.
7. Perform Triangle-subdivision of the unclassified neighbor cells of these border cells.
8. Merge the triangles and assign cluster_id.
9. Repeat steps 4 through 9 till all cells are classified.

The process of forming the coarse cluster starts by considering the cell, say p_i, with the maximum density from the sorted list. From p_i, it initiates the process of expansion by considering its neighboring cells p_{ij} (where cell p_{ij} is the j^{th} neighbor of p_i) depending upon two conditions which are

1. If p_{ij} is not a member of any of the coarse clusters already formed, and
2. The ratio of the densities of the cells p_i and p_{ij} is more than some threshold β' (a user defined input parameter).

Let $d_n(p_i)$ and $d_n(p_{ij})$ denote the densities of p_i and p_{ij}, respectively. Then p_{ij} will merge with p_i, if $\beta' \leq d_n(p_i)/d_n(p_{ij})$. The cells that satisfy the conditions given above are merged to form the coarse clusters. The process of coarse cluster formation continues from p_{ij} in the same way until no neighboring cells p_{jk} of p_{ij} satisfy the condition. The process then backtracks to p_{ij} and restarts with the next neighbor

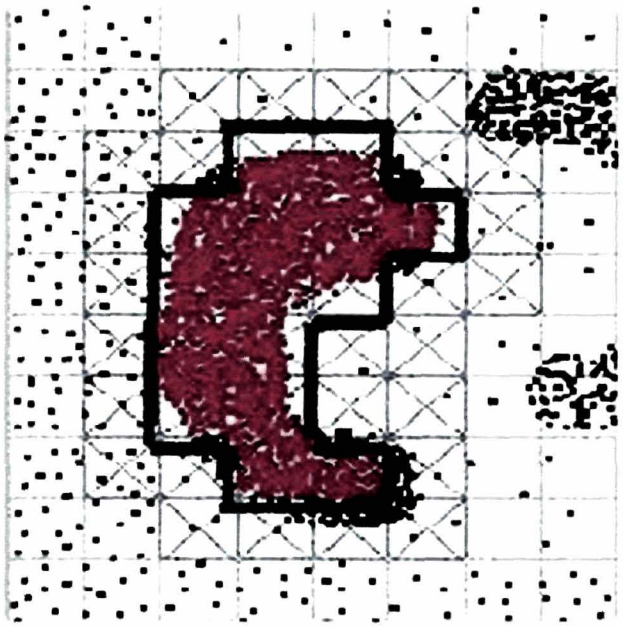


Figure 3.4: Triangle-subdivision of grid cells

cell of p_{ij} which has not already been processed. The coarse cluster formation continues recursively until no more cells satisfy the density confidence condition of a cell.

This coarse cluster is an approximation of the cluster with the maximum density. The cells falling this particular coarse cluster are classified with the same cluster_id. The process then checks the neighbors of the last formed coarse cluster. If any of its neighbors is an unclassified useful cell, both the cell in the coarse cluster as well as the unclassified neighbor cell are triangulated. Suppose p_m is a cell of the coarse cluster last formed and cell p_i is one of its unclassified useful neighbor cells. Then both p_i as well as p_m are triangulated (shown in Figure 3.4). During triangle-subdivision, a particular grid cell is divided into four triangles.

Each of the triangles $T_{p_{iu}}$ inside the cell p_i is verified for the following cases:

Case 1 : If $T_{p_{iu}}$ has a neighbor triangle $T_{p_{mv}}$ which is a part of cell p_m that belongs to a coarse cluster, their densities $d_n(T_{p_{mv}})$ and $d_n(T_{p_{iu}})$ are compared for the density confidence condition of a triangle given as $\beta'/4 \leq d_n(T_{p_{mv}})/d_n(T_{p_{iu}})$. If

this condition is satisfied, triangle $T_{p_{iu}}$ is merged with the triangle $T_{p_{mv}}$ of the coarse cluster and labeled with the same cluster_id as p_m.

Case 2 : If $T_{p_{iu}}$ has a neighbor triangle $T_{p_{iv}}$ which has already been classified and the densities of $T_{p_{iu}}$ and $T_{p_{iv}}$ satisfy the condition given in case 1, $T_{p_{iu}}$ is merged with $T_{p_{iv}}$ and $T_{p_{iu}}$ is classified with the same *cluster_id* as $T_{p_{iv}}$.

The process of triangle merging stops when no more triangles satisfy the density confidence condition of a triangle. Figure 3.4 shows the formation of first coarse cluster and triangle-subdivision of the border cells. The process then starts with the next cell p_2 which is the cell of maximum density in the set of unclassified cells. The process continues recursively merging neighboring cells that satisfy the density confidence condition. Therefore, the coarse cluster formation and triangle-subdivision method are repeated alternately till all the useful cells are classified. The classified cells and triangles now give the distinct clusters and finally the data objects receive the cluster_id of the respective cells and triangles.

The cluster expansion detects embedded and nested cluster structures since after full expansion of a cluster, the algorithm searches for the next candidate seed cell taking into account a variation in density in the dataset. The process starts expanding the new density region till there is again a density variation. This process iterates till all the cells have been classified. The triangle expansion produces finer clustering since the cluster expansion based on cells misses some border objects as seen in Figure 3.3. The expansion based on triangle-subdivision detects the border objects which have been left out by cell based expansion. The final clusters obtained from Figure 3.3 are shown in Figure 3.8. Therefore, the quality of the clusters becomes highly accurate in spite of detecting intrinsic and multi-density clusters.

During clustering, the algorithm considers only grid cells to identify the possible global and embedded clusters and assigns cluster_id accordingly.

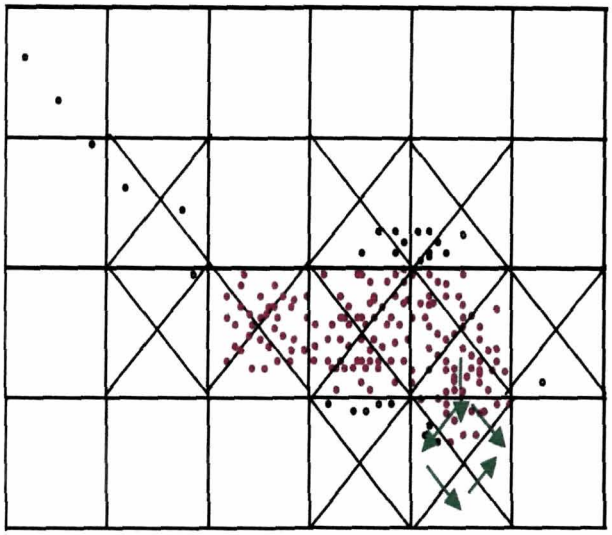


Figure 3.5: The arrows show triangle reachability

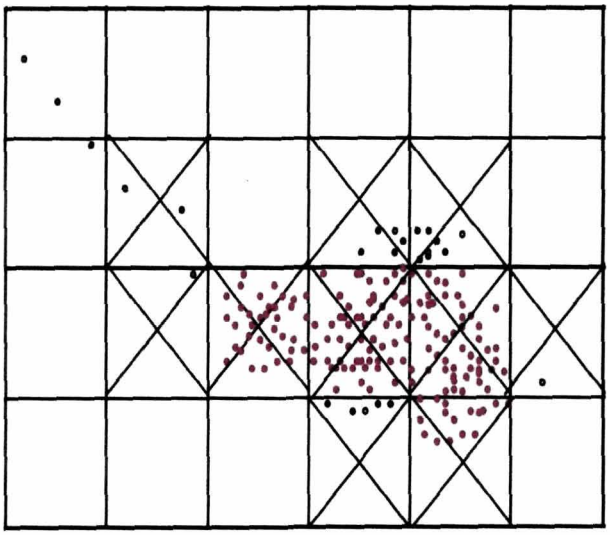


Figure 3.6: Handling of Single linkage problem

Advantages of GDCT

The advantages of the proposed algorithm include embedded cluster detection and handling of single linkage problem. Embedded cluster detection is inherent to GDCT. To understand how GDCT handles the single linkage problem, we consider Figure 3.5. Once the coarse cluster has been formed, the triangle sub-division process starts. For neighbor traversal in triangles there must be at least one common edge between triangles. Two triangles are merged according to Definition 9. As seen in Figure 3.6, a chain of single objects is not be merged as the objects do not satisfy Definition 10. Thus, the single linkage problem which affects DBSCAN does not affect the proposed algorithm.

3.4.1 Complexity Analysis

The partitioning of the dataset into $gr_n \times gr_n$ non-overlapping cells results in a complexity of $O(N)$ where N is the total number of data objects and $N >> gr_n \times gr_n$. Computing density of the cells requires $O(gr_n \times gr_n)$ time. The sorting of cells according to their density results in a complexity of $O((gr_n \times gr_n)log(gr_n \times gr_n))$. The expansion of the coarse cluster requires $O(m)$ time, where m is the average number of cells in an coarse cluster formed and $m << (gr_n \times gr_n)$ in the average case. Cell subdivision into triangles takes place only in case of the border cells of the coarse cluster and its neighboring cells. Assuming there are p border and q neighbor cells, the time complexity is $O(p+q)$. If the number of clusters obtained is k, the overall time complexity for the clustering will be $O(k \times (m + (p+q)))$.

Therefore, the total time complexity is $O((gr_n \times gr_n) + O((gr_n \times gr_n)log(gr_n \times gr_n)) + O(k \times (m + (p+q)))$. Thus the complexity due to partitioning of the dataset into grid cells almost dominates the other components. Therefore, the time complexity becomes $O(N)$ since $N >> (gr_n \times gr_n)$.

Table 3.1: Datasets used

Dataset Name	No. of Objects	No. of	Source dimensions
DS1	4714	2	Generated by us
DS2	6000	2	Generated by us
DS3	8000	2	t8.8k obtained from [KHK99]
DS4	8000	2	t4.8k obtained from [KHK99]
DS5	10000	2	t7.10k obtained from [KHK99]
DS6	8000	2	t5.8k obtained from [KHK99]
DS7	3150	2	Generated by us

3.4.2 Performance Evaluation

All our experiments were conducted on a Pentium IV 2.4GHz computer with 512MB main memory running Windows XP. GDCT was implemented in Java by using Sun's JDK version 1.4.2.

Datasets Used

To evaluate the performance of GDCT, we consider both randomly generated as well as standard synthetic datasets [KHK99]. Table 3.1 gives the number of objects and other related information about the datasets that we use. The synthetic dataset DS1 we have generated is shown in Figure 3.7. The results of DS1 are shown in Figure 3.8. We evaluated GDCT with several other synthetic datasets as well. The clusters obtained for one of them, DS2, is shown in Figure 3.9. GDCT was also applied to the Chameleon t8.8k, t4.8k and t7.10k datasets [ABKS99] and the results obtained are shown in Figures 3.10, 3.11 and 3.12, respectively. The result obtained for t5.8k dataset is shown in Figure 3.13. From these experimental results, it is evident that GDCT is highly capable of detecting intrinsic as well as multi-density clusters. We also observe that the clustering behavior is dependent on the threshold β' which we varied in the interval [0.5, 0.7].

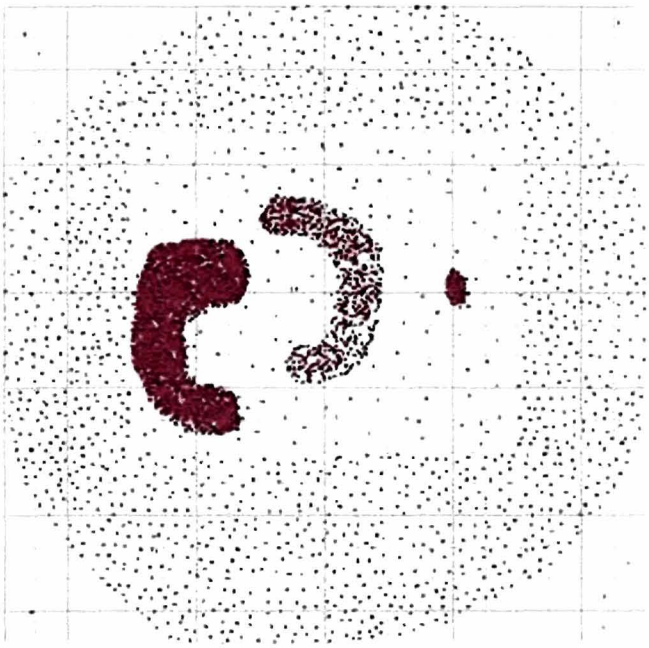


Figure 3.7: Synthetic Dataset DS1

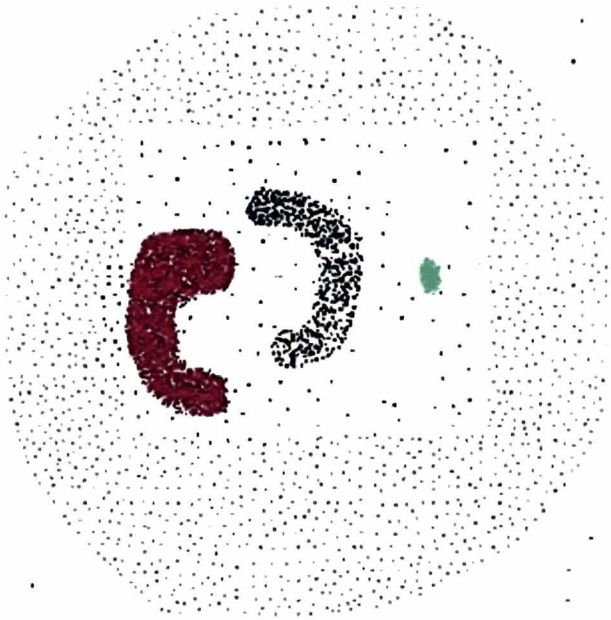


Figure 3.8: Final five clusters in DS1

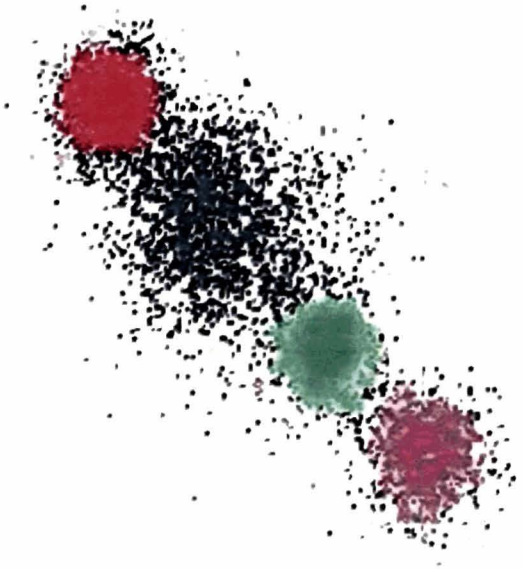


Figure 3.9: Final cluster result of DS2

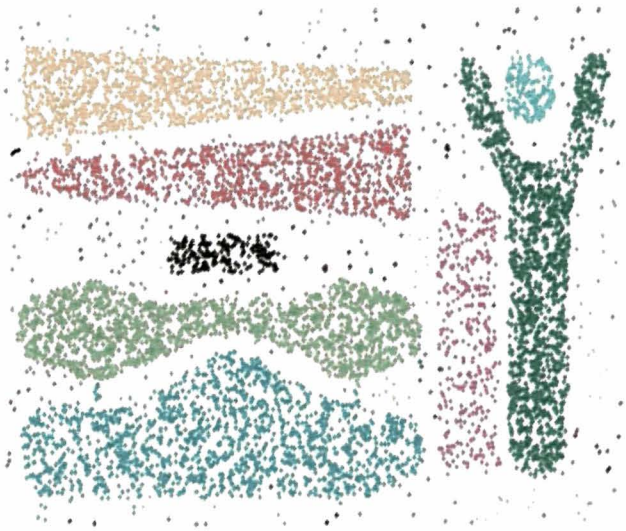


Figure 3.10: Clusters obtained from DS3

Figure 3.11: Clusters obtained from DS4

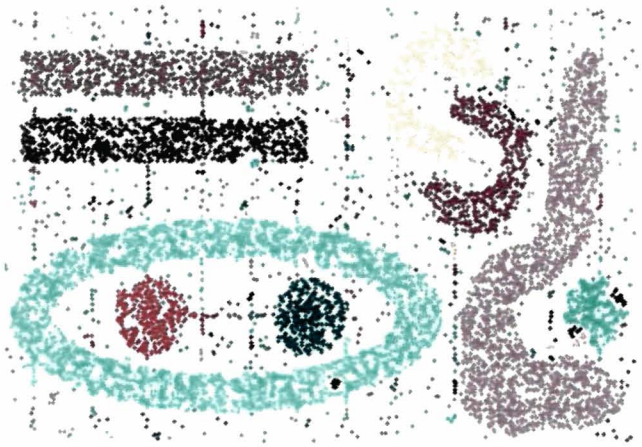


Figure 3.12: Clusters obtained from DS5

Figure 3.13: A total of 6 Clusters obtained from DS6

3.4.3 Performance Comparison

To compare the quality of clustering by GDCT with other relevant algorithms, we used two other clustering techniques (VDBSCAN and DVBSCAN), which are also capable of detecting density varied clusters. For VDBSCAN, we used the K-dist plot [EKSX96] to find the ε values corresponding to different density varied regions in the datasets. Whenever we detected a sharp change (knee) in the K-dist graph, we used the corresponding distance value as the ε value of a density region. If more then one knee were detected, it meant that the dataset had density variations in it; in such a situation the distance values of those in the K-dist plot were used as ε values to detect the different density varied clusters. A dataset without much density variation will have a smooth graph. The results of VDBSCAN on Chameleon t8.8k dataset is given in Figure 3.14. For this dataset we used the value $K = 4$ for $Minpts$ in different runs of DBSCAN for the different values of ε. This figure shows two different results of VDBSCAN for different slope differences. We see that VDBSCAN cannot detect all clusters and suffers from the single link effect for the triangular and arbitrary shaped clusters. When, we tried to separate these clusters by using different slope variations to obtain different ε values, the vertical cluster breaks down, and the clusters does not get separated. For other values, the cluster quality further deteriorates. The results of DVBSCAN on Chameleon t8.8k.dat dataset is given in Figure 3.15. DVBSCAN also gives similar results on this dataset as VDBSCAN. Figure 3.16, shows the

results of VDBSCAN and DVBSCAN on Chameleon t4.8k.dat dataset. Figure 3.17 illustrates the clustering by VDBSCAN and DVBSCAN using Chameleon t7.10k.dat dataset. Figure 3.18 presents the results on Chameleon t5.8k.dat dataset. We can see from Figures 3.16, 3.17 and 3.18 that both algorithms can detect all clusters properly. There are small discrepancies in very small region in Figures 3.16 and 3.17 where there were density variations in very small regions inside clusters. But, our new algorithm GDCT can detect all of the clusters in all the Chameleon datasets effectively. Therefore, in terms of arbitrary shaped clusters as well as small density variations and robustness, our method GDCT performs better than its competitors.

A detailed comparison of GDCT with some relevant clustering techniques w.r.t. features such as the number of parameter, structure, complexities is given in Table 3.2. From Table 3.2, we observe that DBSCAN requires two input parameters $MinPts$ and ε, and it cannot detect embedded clusters. OPTICS on the other hand, requires three input parameters $MinPts$, ε and ε'. But, it can detect embedded clusters. However, its performance degrades while detecting multiple nested clusters over large datasets. Again, GDLC and Density-isoline algorithms can detect multi-density clusters but fail to detect intrinsic cluster structures. GDCT requires the number of grid cells, i.e. n and threshold β' as input parameters. In addition, from our experiments we conclude that the threshold β' does not vary significantly with different datasets. GDCT can effectively detect embedded clusters in variable density space as well as multiple nested clusters even in the presence of noise and outliers. GDCT can also use the outlier handling module as discussed in the next section to differentiate between the various types of outliers.

3.4.4 Handling of Outliers

In this section, we establish that GDCT can successfully handle the various types of outliers. Here, we introduce some definitions related to various outlier types.

1. Distinct Inlier: A cell p_i is called a distinct inlier if all its neighbors, p_{ij}, $(j = 1, \cdots, 8)$, belong to the same cluster i.e., all p_{ij} are classified with the same cluster_id.

Figure 3.14: Result of VDBSCAN obtained from t8.8k.dat dataset

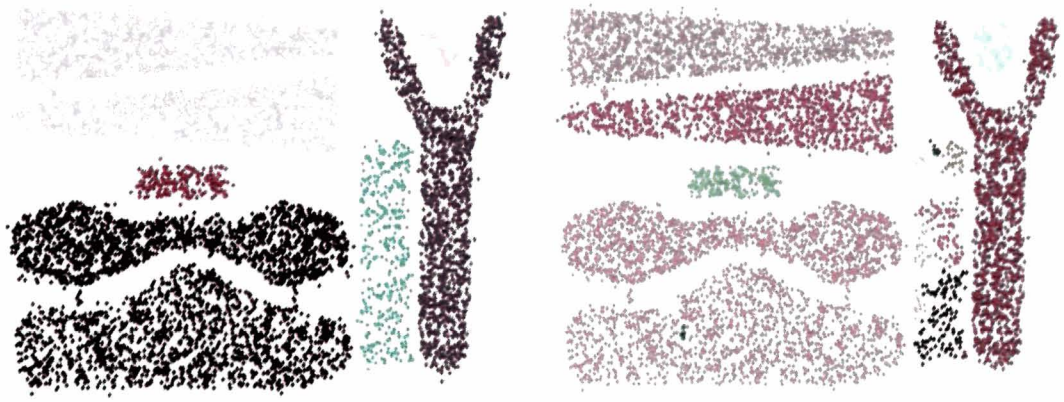


Figure 3.15: Result of DVBSCAN obtained from t8.8k.dat dataset (a) $\varepsilon = 10$, $Minpts = 4$, $CDV = 70$ and $CSI = 20$ (b) $\varepsilon = 8.44$, $Minpts = 4$, $CDV = 70$ and $CSI = 20$

Figure 3.16: Result of VDBSCAN and DVBSCAN obtained from t4.8k.dat dataset (a) $k = 4$ (b) $\varepsilon = 5.2$, $Minpts = 4$, $CDV = 200$ and $CSI = 50$

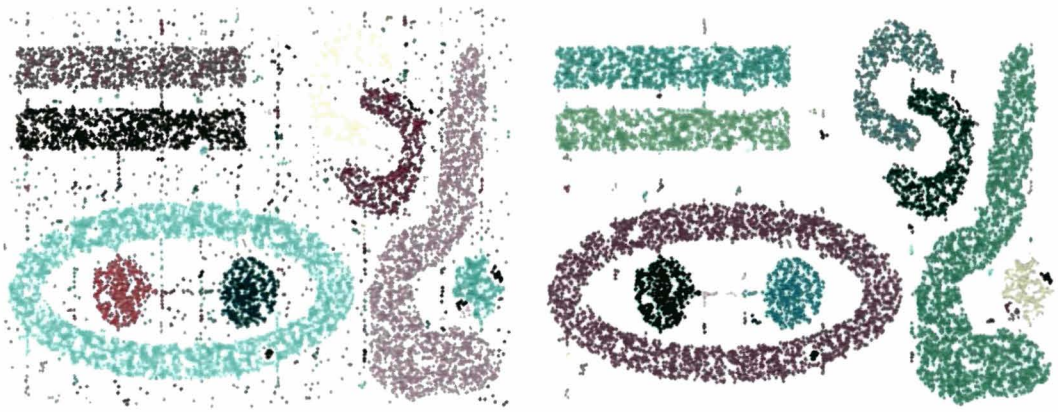


Figure 3.17: Result of VDBSCAN and DVBSCAN obtained from t7.10k.dat dataset (a) $k = 3$ (b) $\varepsilon = 5.9$, $Minpts = 9$, $CDV = 200$ and $CSI = 50$

Figure 3.18: Result of VDBSCAN and DVBSCAN obtained from t5.8k.dat dataset (a) $k = 4$ (b) $\varepsilon = 3.7$, $Minpts = 4$, $CDV = 200$ and $CSI = 50$

2. Distinct Outlier: A cell p_i is a distinct outlier if all its neighbors p_{ij} are unclassified.

   In GDCT, the border cells are triangulated for smoothening the coarse clusters. The following definitions intrduce the other types of outliers helpful for establishing outlier detection capability of GDCT.

3. Border Inlier Cell: A classified cell, p_i, is a border inlier cell if at least one of its neighbor cells is classified with a different cluster_id or is unclassified.

4. Border Inlier Triangle: A classified triangle, T_{p_i}, is a border inlier triangle if at least one of its neighbor cells is classified with a different cluster_id or is unclassified.

5. Border Inlier: A border inlier is the set of all border inlier cells and triangles.

6. Group Outlier: If the number of objects in a cluster is less than a minimum threshold ξ' (where ξ' is an input parameter), the objects are classified as group oulier.

7. Line Outlier: A cell $p_i \in L$, where L is a line outlier if

   (a) $\mid L \mid \geq \gamma$ i.e., L consists of atleast γ cells,

   (b) $1 \leq \mid p_{ij} \mid \leq \eta$ where $\eta = 2$, and

(c) if $\mid p_{ij} \mid = \eta$, then let $p_{ij} = \{p_{i1}, p_{i2}\}$ and $p_{i1} \cap p_{i2} = p_i$ and p_{i1}, p_{i2} are not neighbors to each other.

8. Single Linkage Outlier: A single linkage outlier, S, is defined as follows:

   (a) $\mid S \mid \geq \rho'$, where $\rho' = 5$, i.e., S consists of atleast ρ' cells say $S = \{p_1, p_i, p_j, p_k, p_5\}$,

   (b) $p_1.cluster\_id \neq p_5.cluster\_id$, and

   (c) $\forall p_j; p_j \neq \{p_1, p_5\}$, $\mid p_{jf} \mid = \eta$ (p_{jf} is f^{th} neighbor of p_j), and let $p_{jf} = \{p_i, p_k\}$ i.e., $p_i \cap p_k = p_j$ and p_i, p_k are not neighbors to each other.

From our experiments, we find that $\gamma = 3$ and $\eta = 2$. Line outliers for triangles are also obtained analogously. Similarly, triangles may also contribute to single linkage outlier. It is a chain of cells and triangles satisfying the line outlier property but the two ends of the line are classified with different cluster_ids.

Apart from detecting embedded as well as varying density clusters, GDCT uses the outlier handling module based on the above definitions to handle the special forms of outliers mentioned above. Once GDCT obtains clusters, the outlier detection process starts. The outlier detection algorithm is given in Figure 3.19. The algorithm starts by checking all the $gr_n \times gr_n$ grid cells. If any of the cells satisfies the distinct outlier condition as given earlier, all objects in it are marked as distinct outlier. Otherwise, the check_cluster_outlier function is called to find all the different forms of inliers and outliers as given in Figure 3.20. When the outlier detection algorithm is used on clusters obtained by GDCT on DS7 (as given in Figure 3.21), we obtain the different types of inliers and outliers as shown in Figure 3.22. Figure 3.22, illustrates the different cases of outliers and inliers graphically by using a different color for each case.

3.5 Discussion

This chapter presents a clustering method based on a grid-density approach. Our method is able to detect global as well as embedded clusters. An outlier handling

check_outlier(set of cells ($gr_n \times gr_n$))

```
  id = 0;
  For cell i=1 to gr_n × gr_n do
    status = check_distinct_outlier(i);
    if status == 1
      id = distinct_outlier;
      Mark cell_i.id = id;
      Mark all objects in cluster cell_i with id;
    else
      check_cluster_outlier(cell_i.cluster_id);
    End if
  End for
End
```

Figure 3.19: Algorithm for outlier detection

module is also presented for detection of different types of outliers. Experimental results using several standard synthetic datasets are reported to establish the superiority of the algorithm. In this chapter, we have only considered two-dimensional spatial objects. Therefore, there is scope for the enhancement of GDCT to detect clusters in higher dimensional datasets with minor modifications. In the next chapter, a technique for clustering satellite image data is introduced.

check_cluster_outlier(cl_id)

```
id = 0;
if check_group_outlier(cl_id) == 1
   id = group_outlier;
   if check_line_outlier(cl_id) == 1
      id = line_outlier;
   End if
End if
else
   if check_line_outlier(cl_id) == 1
      id = line_outlier;
   End if
   else
      if check_distinct_border_inlier(cl_id) == 1
         id = distinct_border_inlier;
      End if
      else
         id = border;
         if id == border
            if check_single_linkage(cl_id) == 1
               id = single_linkage
            End if
         End if
         Mark all objects in cluster_cl_id with id;
      End else
   End else
End else
End
```

Figure 3.20: Algorithm for checking the different cases

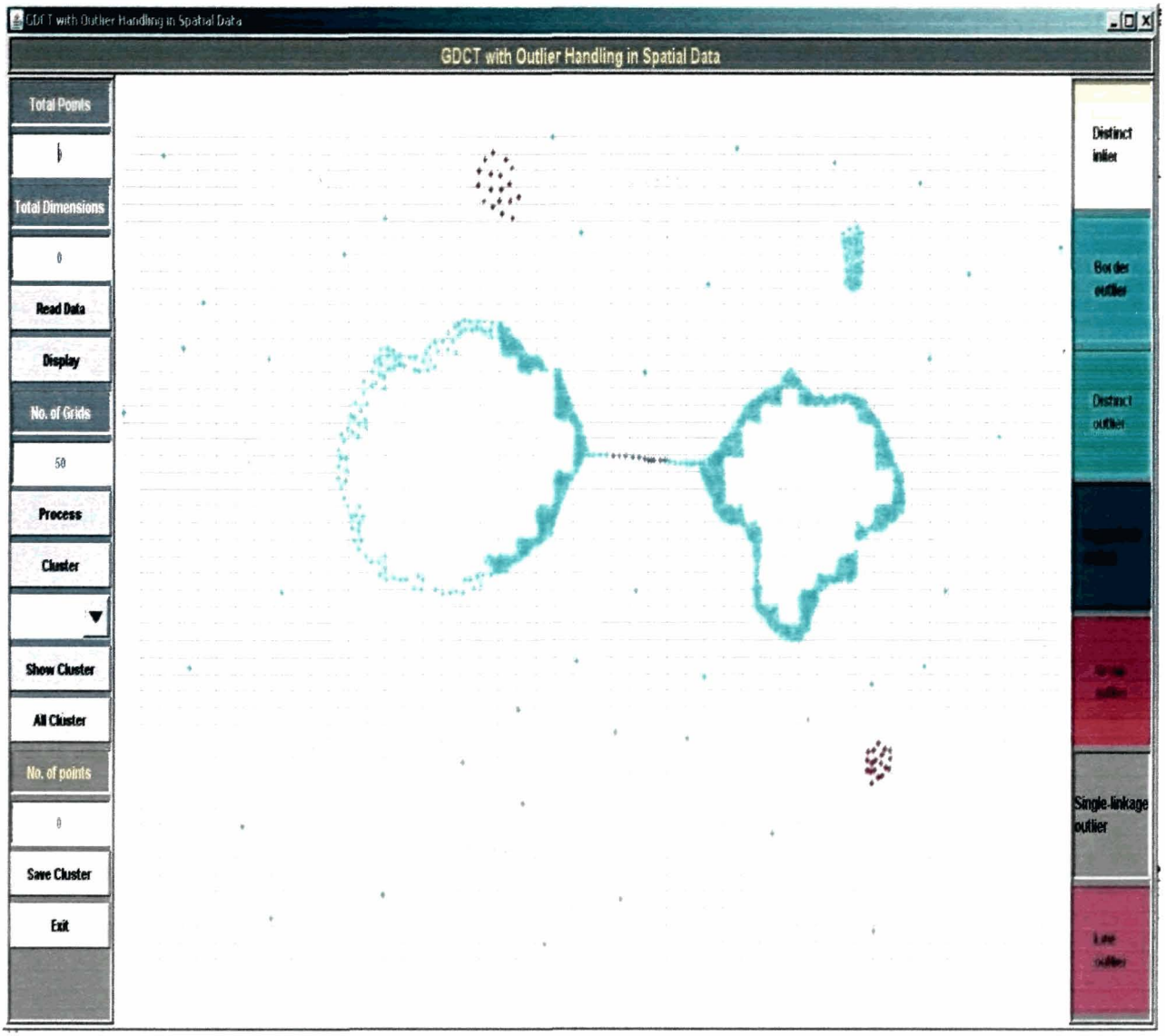


Figure 3.21: Result of GDCT obtained on a synthetic dataset generated by us

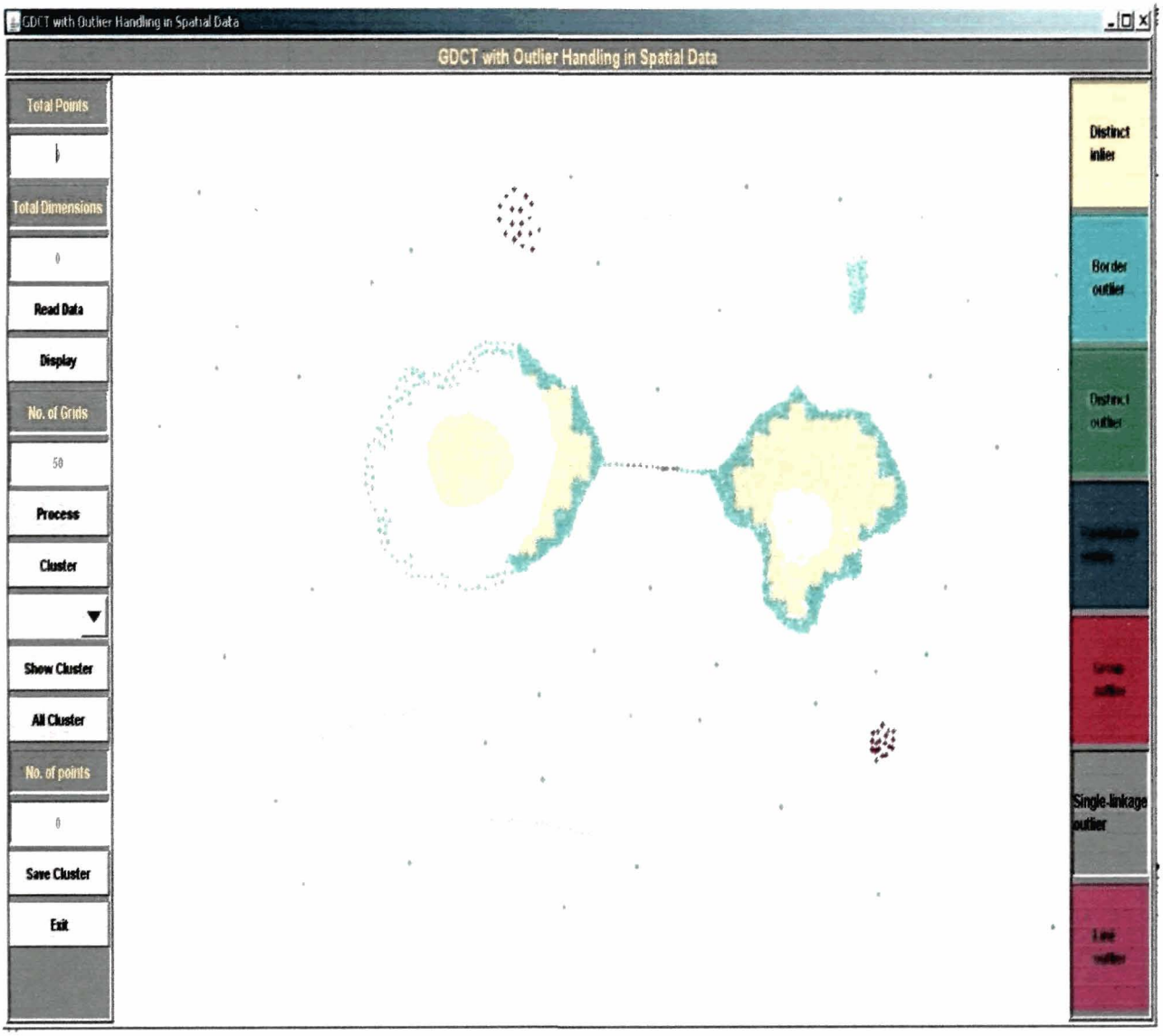


Figure 3.22: Result of outlier detection

Table 3.2: Comparison of GDCT with different clustering algorithms

Algorithms	*No of Parameters*	Optimized For	Structure	*Multi-Density Clusters*	*Embedded Clusters*	*Complexity*	*Noise Handling*	*Order Independence*
CHAMELEON	3 (k-nearest neighbor MINSIZE, α^c)	Small datasets	Arbitrary	Yes	No	$O(N^2)$	Yes	Yes
DBSCAN	2 ($MinPts$, ε)	Large datasets	Arbitrary	No	No	$O(NlogN)$ using R^* tree	Yes	Yes
OPTICS	3 ($MinPts$, ε, ε')	Large datasets	Arbitrary	Yes	Yes	$O(NlogN)$ using R^* tree	Yes	Yes
EnDBSCAN	3 ($MinPts$, ε, ε')	Small datasets	Arbitrary	Yes	Yes	$O(NlogN)$ using R^* tree	Yes	Yes
EDBSCAN	4 ($MinPts$, ε, δ, τ)	Large datasets	Arbitrary	Yes	Yes	$O(NlogN)$ using R^* tree	Yes	Yes
DDSC	3 ($MinPts$, ε, α_D)	Large datasets	Arbitrary	Yes	Yes	$O(NlogN)$ using R^* tree	Yes	Yes
VDBSCAN	3 ($MinPts$, ε, ε')	Large datasets	Arbitrary	Yes	Yes	$O(NlogN)$ using R^* tree	Yes	Yes
DVBSCAN	3 ($MinPts$, ε, CDV_α, CSI_λ)	Large datasets	Arbitrary	Yes	Yes	$O(NlogN)$ using R^* tree	Yes	Yes
LDBSCAN	3 ($LOFUB$, pct, $Minpts_{LOF}$, $Minpts_{LDBSCAN}$)	Large datasets	Arbitrary	Yes	Yes	$O(NlogN)$ using R^* tree	Yes	Yes
GDCT	2 (gr_n,β')	Large datasets	Arbitrary	Yes	Yes	$O(N)$	Yes	Yes

Chapter 4

Grid-Density based Clustering for Pan-Chromatic and Multi-Spectral Satellite Data

High resolution and high dimensional satellite images contain clusters of different sizes, shapes and densities in addition to contain huge amount of data. Due to these reason, most algorithms for clustering satellite data sacrifice the correctness of their results to achieve better processing time. The processing time is greatly influenced by the amount of information that needs to be processed. In this chapter, we propose two grid density based clustering methods for detecting clusters present in satellite images. Each method is comprised of two phases. In the first phase, a grid density based technique is used to determine the initial (or rough) clusters. This phase can detect the overall clusters but the cluster boundaries are not smooth and are jagged in appearance due to the inherent problem with grid clustering. Therefore, a second phase is incorporated to smoothen the cluster borders. Experimental results using several real-life datasets are reported to establish the efficiency of the clustering methods.

4.1 Introduction

Clustering is the organization of a dataset into well separated partitions or clusters with respect to a similarity measure. The main characteristic of the clusters discovered is that they conserve the homogeneous property within a cluster *i.e.*, data objects within the same cluster are more similar than the data points belonging to different clusters [HK06]. A high dimensional satellite image is a remotely sensed image of the earth's surface. Such an image is a collection of a huge amount of pixel data. In a high dimensional satellite image, each pixel represents an area on the earth's surface. Multi-spectral images constitute the main type of images acquired by remote sensing. It is a technology originally developed for space-based imaging to capture light from frequencies beyond the visible range (e.g., infrared). It enables the extraction of additional information that the human eye fails to capture with its receptors for red, green and blue. A multi-spectral satellite image is a digital image of multiple bands where each band represents a particular wavelength of light. Segmentation or clustering a multi-spectral satellite image is a complex problem that has been pursued for a long time. Clustering satellite images is the process of discovering a finite number of non-overlapping and meaningful regions or clusters in the image data space. Remotely sensed satellite images mainly consist of objects (regions) such as vegetation, water bodies, concrete structures, open spaces and habitation. The regions are separated from one another due to their different reflectance characteristics. This leads to a wide variety of clusters of different sizes, shapes and densities.

There are two fundamental properties of a pixel value: (i) discontinuity– discontinuities between gray level regions can be used to detect isolated points and contours within an image, and (ii) similarity– decision criteria can be used to separate clusters in an image based on the similarity of the pixel values. Based on these two fundamental properties, several image segmentation methods have been developed. Clustering approaches are based on the second property. Due to the presence of a huge amount of data in satellite images, there is utmost need for a good clustering algorithm which can efficiently detect clusters. A good clustering technique for satellite images (i) should have minimal number of input parameters, (ii) be able to detect arbitrary shaped clusters accurately, and (iii) demonstrate good efficiency on

large datasets.

In this chapter, we present two new grid density based clustering methods each of which work in two phases: Phase I identifies the rough clusters and Phase II smoothens the cluster boundaries detected in Phase I. Phase I is same for the methods while Phase II differs. Experimental results are reported to establish that the proposed methods can determine all the classes present in any satellite image data effectively and dynamically.

Each pixel in the image is represented by a 5 dimensional tuple: $(x,\ y,\ h,\ s,\ i)$, where x and y are the pixel's coordinates, h the hue, s the saturation and i the intensity of the pixel. In Phase I, the image data is divided into equal sized grids based on the values of the $(x,\ y)$ coordinates. The maximum occurring hue value among all the pixels is found and the grid cells whose pixels have this hue value become the seed for cluster expansion. The grid cells are then clustered by a topological search algorithm. Grid cells with similar hue are merged to form the clusters. When no more expansion is possible, the method checks for unclassified grid cells with hue values similar to that in the most recently formed cluster. Cluster expansion starts as before with these cells tagged with the same cluster id as the recently formed cluster. This process continues iteratively until no more grid cells satisfy the given condition. The method then restarts with the next maximum occurring hue from among the unclassified grid cells. When no more cluster expansion is possible the method is terminated. Thus, we obtained all the clusters present in the image data. After obtaining the rough clusters, we concentrate on smoothening the boundaries of the clusters in Phase II. The proposed methods do not require any prior knowledge or a set of initial seeds to form the cluster centers. Neither does the number of clusters play any role in the clustering process. The proposed methods were tested on large number of multi-spectral satellite image data and the cluster results were found very satisfactory. A major advantage of this method is its simplicity. In addition, there there is no need to make initial guesses about the cluster centers or the number of clusters.

The rest of the chapter is organized as follows. We discuss a few well-known clustering algorithms in Section 4.2. Section 4.3 provides the basics of our grid-density based clustering methods. In Section 4.8 we present the experimental evaluation of the effectiveness and efficiency of the proposed methods. Finally, Section 4.9 concludes with a concluding discussion of our work.

4.2 Related Work

In this section we discuss a few methods that have been used for clustering satellite image data.

4.2.1 Clustering Satellite Images

K-means [McQ67] and ISODATA [BH67] are two popular algorithms widely used for detecting clusters in satellite images. However, k-means depends heavily on user-supplied parameters such as the number of clusters and initial cluster centers. Result can be made more data dependent and the need to provide the number of clusters may be relaxed to some extent if one uses the ISODATA algorithm. But, the main problem with these algorithms is that they require several parameter values to be supplied by the user. Hence, the performance of these clustering algorithms is very much dependent on the parameter values, the chosen measure of similarity and the method used for identifying clusters in the data.

In [Yam98a], a robust clustering technique for multi-spectral satellite images was developed. The observed image data were assumed to come from a mixture of multivariate normal densities and the number of different densities present in the dataset was assumed to be known. In the clustering technique the parameters were tentatively estimated by a multi-dimensional histogram and a minimum distance classification method. The EM (Expectation Maximization) algorithm improved the estimates of the mixture of density parameters recursively. The satellite image classification was carried out by the conventional maximum likelihood method with the estimated parameters. The method is robust for noises and gives stable classification.

The work in [AZM06] segments satellite image data based on the FCM algorithm detects different classes in it.

FCM [Bez81a] is a local search optimization algorithm that converges to a local minimum point. FCM allows an object to belong to two or more clusters with varying degrees of membership. The FCM algorithm attempts to partition a finite collection of elements into a collection of fuzzy clusters with respect to some given criterion. In fuzzy clustering, each point has varius degrees of belonging to clusters, as in fuzzy logic, rather than belonging completely to one cluster. Thus, points on the edge of a cluster may be in the cluster to a lesser degree than points in the center of cluster. FCM is quite effective for image segmentation, but the quality of clusters produced is greatly affected by the proper selection of initial values. FCM has been used separately but it rarely has showed success without combination with another method.

Awad et al. in [ACN09] proposed a method which uses FCM and Hybrid Dynamic GA (HDGA). FCM gets the cluster centers from HDGA before segmenting different types of satellite images. This cooperative approach showed high accuracy in segmenting this type of complex images.

A new approach based on SOMs and FCM is reported in [AN09]. This method uses an unsupervised parameter free approach to segment different types of satellite images successfully. The approach has been applied to both medium and high resolution satellite images.

In [PC02], a simple measure of circular symmetry is used to extract all clusters including sub-clusters which are then used as building blocks to form the final clusters of arbitrary shapes by merging and splitting. This method does not require initial guesses regarding the cluster centers or the number of clusters. On the other hand, it initially considers each data point as a cluster center.

Other approaches to segmentation of remotely sensed satellite images include fuzzy thresholding techniques reported in [PGS00].

Genetic algorithms have been used to classify satellite image data in [BP01], in particular for partitioning different land cover regions with complex/overlapping cluster boundaries. These methods use supervised classification where prior knowledge about the images is essential. Real-coded variable string length genetic fuzzy clustering is used in [MB03a] to classify satellite images. The clusters are automatically evolved to the appropriate number of clusters. A multi-objective genetic optimization algorithm is presented in [BMM07a] to determine cluster centers and the corresponding partition matrix. Fuzzy clustering is modeled using two cluster validity measures that are simultaneously optimized.

An image segmentation technique using M-band wavelet packet frames is presented in [AK07]. Here, unsupervised feature extraction is used to select the appropriate features from the output of the wavelet decomposition. A neuro-fuzzy feature evaluation technique is used to select an optimal set of features. The features thus obtained are used to segment satellite image data. The EM (Expectation Maximization) [Yam98b] algorithm improves the estimates of the mixture of density parameters recursively. The satellite image classification is carried out by the conventional maximum likelihood method with the estimated parameters. The method can tolerate the presence of noise and gives stable classification but is computationally expensive.

In [GyFSlXr09], a remote sensing image segmentation method based on an improved FCM (fuzzy c-means) algorithm is presented. It uses the Mahalanobis distance as proximity measure. This method can solve the problem of selecting the initial cluster centers by combining the Evolving Clustering Method (ECM) with the modified FCM algorithm. The combination enables the FCM algorithm to converge to a global optimal with fewer iterations.

4.2.2 Discussion

Based on our survey, we come to the conclusion that there is no single algorithm that can effectively handle the following three factors at the same time: non-dependency on input parameters, fast processing time and quality cluster detection. Furthermore, mixed pixels (mixels) are present in satellite images. Mixels are not completely occupied by a single and homogeneous object and occur because the pixel size may not be fine enough to capture details on the ground. Fuzzy methods in remote sensing have received growing interest in these situations where the geographical phenomena are inherently fuzzy and consist of mixed pixels. The rest of the chapter presents two satellite clustering methods that address the above mentioned challenges. Each method has two phases and the first phase of both are the same. The difference is in the second phase. In the first phase a grid density based approach is used to detect the initial or rough clusters. For the second phase one of the methods (SATCLUS) uses a partition based approach for refining the rough clusters to obtain the final set of clusters. The other method (GDSDC) uses a fuzzy membership function to obtain the final clusters in the second phase. In the next section we present the foundational material including the basic definitions used by both methods. In Section 4.4, we present the rough clustering phase (Phase I) which is same for both SATCLUS and GDSDC. In Sections 4.5 and 4.6, we present the second phase of SATCLUS and GDSDC.

4.3 Basics of SATCLUS and GDSDC

High resolution and high dimensional satellite images cause difficulty for clustering methods due to the presence of clusters of different sizes, shapes and densities as they contain huge amount of data. Due to this reason, most algorithms for clustering satellite data sacrifice the quality of their results to achieve fast processing time. The time taken is greatly influenced by the amount of information that needs to be processed. In the rest of the chapter, we develop two grid based clustering methods for detecting the clusters present in satellite images. We establish the efficiency of the methods through experimental results. The aim of our clustering methods is to discover clusters in satellite image datasets. In both methods, we regard each pixel

data as a point in space. The image data space is divided into grid cells and the grid cells whose HSI values are similar with respect to neighboring cells (see Figure 3.2 (a)) are merged. Once merging of grid cells according to HSI values terminates, a set of rough clusters is obtained. The border cells in a cluster are found and the clustering proceeds at the pixel level to obtain the finer clustering of the dataset. Based on [EKSX96], we introduce some definitions, which are used in the proposed methods. The basis of the definitions has been taken from [SDB08].

Definition 13. Density of a cell: It is the number of pixels within a particular grid cell.

Definition 14. Difference value of a pixel: It is the distance between the HSI values of a pixel w.r.t. the seed pixel. If it is within the range of certain threshold θ, the difference value is considered 1 else 0.

The distance may be any of the distance measures discussed in Chapter 2. We have used the Mahalanobis distance since it gives better results in our experiments.

Definition 15. Population count: The population count of a Grid cell is defined as the number of ones in each grid cell.

Definition 16. Population-object ratio: It is defined as the ratio of the population count and cell density of a grid cell.

$$Population\_object\_ratio = \frac{population\_count}{cell\_density} \tag{4.1}$$

Definition 17. Confidence in a cell: A current cell is said to have confidence in one of its neighbors if the difference of their population-object ratio is greater than or equal to some threshold ω, where ω is a user input. Confidence plays an important role in cluster formation. Two cells p and q are merged into the same cluster if the following condition is satisfied:

$$\omega \leq \left| \frac{P_o(p)}{P_o(q)} \right|$$

where $P_o(p)$ represents the population-object ratio of a particular cell p.

Definition 18. Reachability of a cell: A cell p is reachable from a cell q if p is a neighbor cell of q and cell p satisfies the confidence condition w.r.t. cell q.

Definition 19. Rough cluster: A rough cluster is defined to be the set of points in the set of reachable cells. A rough cluster C_i w.r.t. ω is a non-empty subset satisfying the following condition:

> $\forall p, q$: if $p \in C$ and q is reachable from p w.r.t. ω, then $q \in C$, where p and q are cells.

Definition 20. Border cell: A cell p is a border cell if it is part of a rough cluster C_i and at least one of its neighbors is part of another rough cluster C_j.Gene Based Clustering Algorithms

Definition 21. Noise: Noise is simply the set of points in the cells that are not in any of its clusters. Let $C_1, C_2, \cdots C_k$ be the clusters w.r.t. ω. Then

> $Noise = \{no\_p \mid p \in n \times n, \forall i : no\_p \notin C_i\}$ where no_p is the set of points in cell p that are not in any of the clusters C_i $(i = 1, \cdots, k)$.

4.3.1 Confidence in a Cell

The confidence in a given set of cells reflects the general trend of that set. If the information of one cell is abnormal compared to the others it is not be included in the set. Similarly, each cell has a level of confidence on each of its neighbor cells. If the confidence of a current cell on one of its neighbor cell does not satisfy the confidence condition, that neighbor cell is not included into the local cluster area. On the contrary, if it satisfies the condition, we treat the neighbor cell as a part of the local cluster area and merge the cell with the cluster area to form the rough cluster. This method has the ability to recognize the local clusters in the data space in the presence of embedded clusters also.

In light of the above definitions, the following lemmas are trivial.

Lemma 4. Let C_1 and C_2 be two rough clusters w.r.t ω and let p be any cell in C_1 and q be any cell in C_2. Then, for a cell r, if r is reachable from p w.r.t ω, r is not reachable from q w.r.t ω, i.e., $r \in C_1$ and $r \notin C_2$.

Lemma 5. Let C be a set of clusters w.r.t ω and let p be a cell corresponding to noise points. Then,

$\forall p : p$ is not reachable from any cell in C i.e. $p \notin C$.

Both our algorithms, SATCLUS and GDSDC, have the same first phase and starts by dividing the image space into $gr_n \times gr_n$ non-overlapping square grid cells, where n is a user input. Each image pixel is mapped to its corresponding grid cell. It then calculates the density of each cell and converts the RGB values of each pixel to its corresponding HSI values. The methods uses the cell density in the grid structure and clusters the data points according to the densities in the surrounding cells.

Both SATCLUS and GDSDC are divided into two phases. In the first phase a rough clustering of the image space is obtained and the second phase deals with cluster smoothening for quality cluster identification.

4.4 Phase I: Rough Clustering phase of SATCLUS and GDSDC

The maximum hue value in the grid is selected and an arbitrary pixel with this hue value becomes the seed for cluster initiation. An example is shown in Figure 4.1 (a) where the shaded pixel is the seed. Each grid cell contains 4 pixels. The difference of the HSI values of the remaining pixels with this seed is calculated. If the difference of the HSI values of a particular pixel and the seed pixel is less than some threshold θ, that corresponding pixel difference value for that pixel becomes 1 else 0. The image is converted into a 0-1 matrix as shown in Figure 4.1 (b). The population count of each grid cell is computed and the corresponding population-object-ratio calculated. The clustering process now starts with the grid cell with the highest population-object ratio as shown by the shaded grid cell in Figure 4.2. The remaining cells are then clustered iteratively according to their population-object ratio values, thereby building new clusters or merging with existing clusters. Only the cells adjacent to a cluster can be merged. A neighbor search is conducted, starting at the highest population-object-ratio value grid-cell and inspecting adjacent cells. If a neighbor

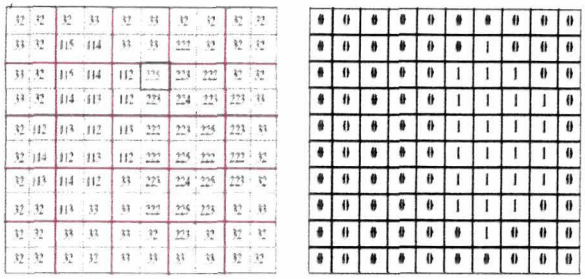


Figure 4.1: a)An example image with 5×5 grids and the hue values for corresponding pixels, and b) A 0-1 matrix obtained from the difference value w.r.t. seed

cell which satisfies the density confidence condition on a cell is found, then the neighbor cell is merged with the current cell and the search proceeds recursively with this neighbor cell. This search is similar to a graph traversal where the nodes represent the cells and an edge between two nodes exists if the corresponding cells are adjacent and satisfies the confidence condition on a cell.

Merging of cells stops when no more cells satisfy the confidence condition on a cell. The process then starts the next cluster from the set of unclassified cells with the maximum hue pixel value. The process continues recursively merging neighboring cells that satisfy the confidence condition on a cell. This process of merging cells and selecting seeds is repeated until all the useful cells have been classified. The classified cells represent the set of rough clusters and finally the pixels receive the cluster_id of the respective cells.

The algorithm for rough clustering (Phase I) includes the following steps.

1. Create the grid structure.
2. Compute the density of each cell.
3. Convert the RGB values of each pixel into their HSI values.
4. Identify the cell with the maximum hue value as the seed.
5. Calculate each pixel's difference value w.r.t. the seed.

0	0	0	1/4	0
0	0	2/4	4/4	1/4
0	0	2/4	4/4	2/4
0	0	2/4	4/4	1/4
0	0	0	1/4	0

Figure 4.2: Population-object ratio of each grid cell.

6. Compute population count of each grid cell and calculate the corresponding population-object ratio.

7. Traverse the neighbor cells starting from the grid cell with the highest population-object ratio value.

8. Merge the cells and assign cluster_id.

9. Repeat steps 5 through 9 till all the cells are classified.

Figure 4.3 (b) shows the result of the rough clustering phase of an example image shown in Figure 4.3 (a). The rough clusters obtained are grainy in nature. This is a drawback of grid based method. To obtain clusters with smooth borders, the border cells are detected and re-clustered using either a: (i) Partitioning approach or (ii) a Fuzzy approach.

Since Phase I is a sampling-biased technique, the clusters formed may not be accurate near their boundaries. This may be due to the presence of the mixed pixels as well as the size of the grid. So, to address these problems, we concentrate on smoothening the boundaries of the clusters in Phase II. In the cluster smoothening step, the number of clusters is already known since it is given by the number of rough clusters so formed. The border cells are detected according to Definition 20. Phase II may be executed in two different ways depending on which technique we choose. The algorithm for Phase II of SATCLUS is discussed in Section 4.5 and the algorithm for Phase II of GDSDC is presented in Section 4.6.

4.5 Phase II: Hard Clustering Approach for SATCLUS

The first method (SATCLUS) uses partitioning to reassign the border objects of the rough clusters to their appropriate clusters. In Phase II, the border cells of the rough clusters obtained are detected as shown in Figure 4.3 (c) for the image of Figure 4.3 (a). Once the border cells have been found, clustering process starts at the pixel level.

Suppose, the number of rough clusters created in the first phase of clustering is k. Now, the pixels in the border cells are checked for their re-assignment to clusters to improve the quality of clusters formed. The k rough clusters each has one seed pixel. Let x be a pixel in a border cell. The distance of x with each of the k seeds is calculated and x is assigned to that cluster from which it has the least distance w.r.t. the seed. This process is repeated for all pixels in the border cells. The final set of clusters obtained after processing the border pixels using partitioning in Phase II is shown in Figure 4.3 (d).

The algorithm for the cluster smoothening is given below:

Input: q border cells, k seeds corresponding to the k rough clusters obtained from Phase I

1. Start with an arbitrary border pixel x.
2. Find the distance of x to each of the k seeds.
3. Assign x to the cluster to which it has minimum distance w.r.t. the seed.
4. Repeat steps 1 to 4 till all border pixels have been reassigned.

In satellite images there is always the possibility of the presence of mixed pixels. The handling of such pixels is very important and challenging. Our next technique (GDSDC) deals with a fuzzy approach that helps in smoothening of the rough clusters and handling the mixed pixels.

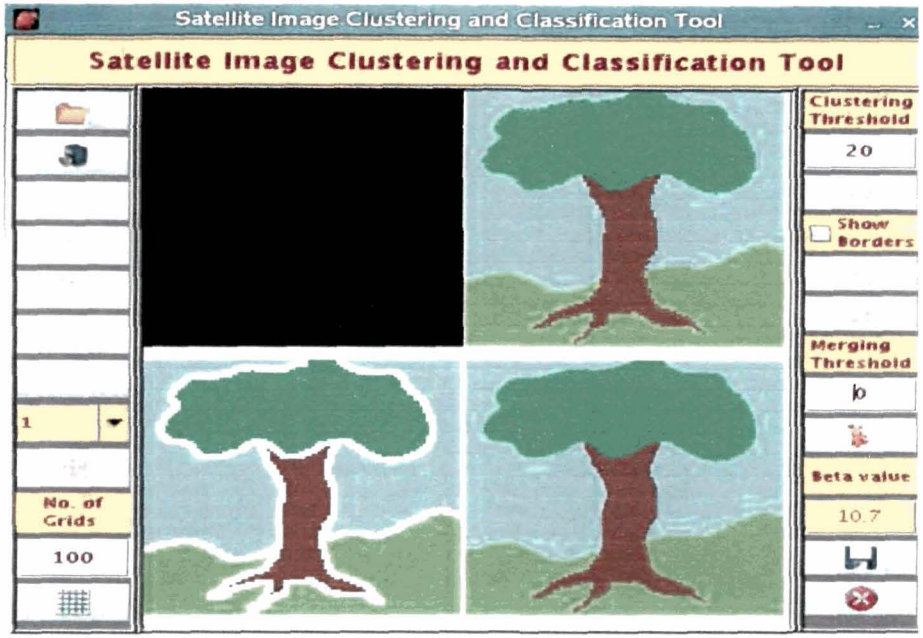


Figure 4.3: a) An example image with its grid structure, b) The four rough clusters, c) Clusters along with their borders, and d) The final clusters

4.6 Phase II: Soft Clustering Approach for GDSDC

Mixed pixels (mixels) occur in a satellite image because the pixel size may not be fine enough to capture details on the ground. Therefore some pixels are not completely occupied by a single and homogeneous object and fuzziness occurs due to the presence of mixels. Fuzzy methods are appropriate in remote sensing since geographical data are inherently fuzzy. Among the various types of fuzzy membership functions available in the literature, the function reported in [Bez81a] is used in GDSDC to classify the border pixels, which are common in satellite data, due to its effectiveness in terms of accuracy.

4.6.1 Mixed Pixel Handling

The spatial resolution of satellite sensor systems imaging the earth is coarser than the sizes of objects on ground. One pixel in the satellite image usually covers more than one object on the ground. Pixels in the image covering more than one object

on the ground are mixed pixels. Mixed pixels present in a satellite image greatly affect the quality of clusters produced since these pixels have a signature representative of more than one cluster (as with boundary pixels). Figure 4.4 (a) shows a satellite image of resolution $30m \times 30m$ which means each pixel in the image represents an area of $30m \times 30m$ on the ground. This image is reproduced from *http://clear.uconn.edu/projects/landscape/images/measuring/aerialgrid.gif*. In Figure 4.4 (a) each grid cell represents a pixel and each pixel may not contain all homogeneous objects. As an example, consider pixel 3. We see that this particular pixel represents a house, cars, trees, fields, shrubs and a road. Therefore pixel 3 itself consists of many non-homogeneous objects. Thus, it represents more than one object on ground and is a mixed pixel. Different types of mixed pixels are shown in Figure 4.4 (b). This image is reproduced from *http://wgbis.ces.iisc.ernet.in/energy/paper/TR-111/chapter4_clip_image002.jpg*. The first case is that of a *sub-pixel* where a single pixel represents information from more than that of one sub-pixel. In addition, the objects are not homogeneous at sub-pixel level. The second case is *boundary pixel*. In this case we see that mixed pixel occurs at the boundary between different objects. In the third case the intensity values change gradually. This is known as *intergrading*. The fourth case is that of a *linear sub-pixel* (i.e., different objects are aligned linearly at sub-pixel level) as given in Figure 4.4 (b). Traditional clustering techniques can find clusters present in a satellite image, they are prone to misclassify mixed pixels. Even with the improved resolution of the satellite images, the mixed pixel problem remains. Therefore, to handle this problem, some researchers incorporate fuzzy classification allowing a mixed pixel (or border pixel) to belong to two or more clusters. FCM [Bez81a], discussed previously, attempts to handle the mixed pixel classification problem using the principles of fuzzy sets to evolve a set of prespecified number of partitions with minimum intra-cluster dissimilarity. However, FCM suffers from two common drawbacks: (i) it requires the number of clusters, and (ii) it often gets stuck at suboptimal solutions based on the initial configuration of the system. To address these challenges, this section presents a fuzzy border pixel classification approach that can effectively solve the mixed pixel problem as well as smoothen the bordering regions of the clusters obtained by the grid-density based clustering algorithm discussed in Phase I. The proposed approach has the following

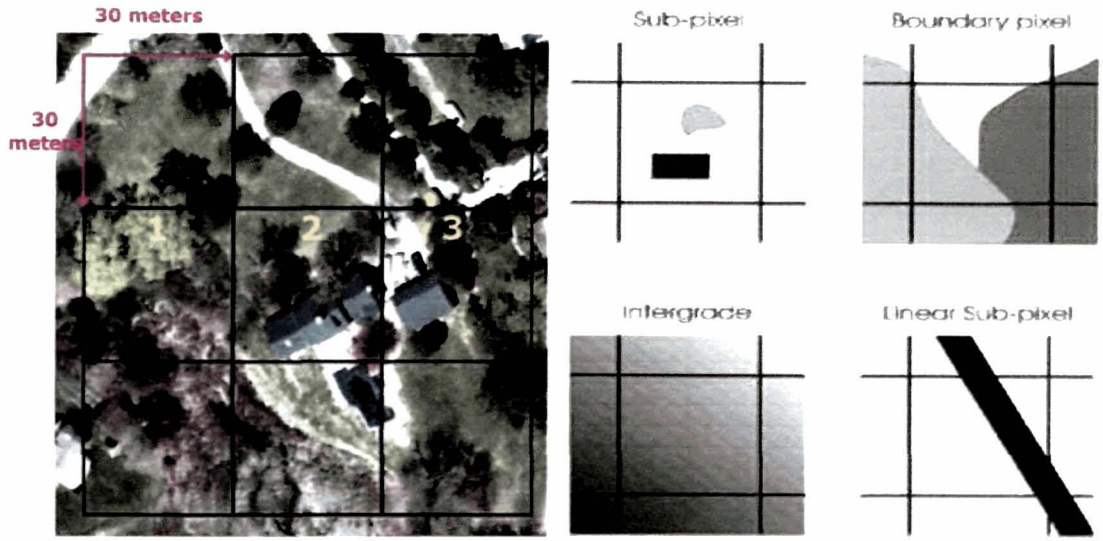


Figure 4.4: a) Mixed Pixels b) Different cases

distinguishable features.

- Unlike FCM [Bez81a], it does not require the number of clusters as a input parameter.
- Unlike the other density-based approaches, it works over grid space which makes the method faster.
- Unlike the other grid-based approaches, it can detect clusters with diagonal boundaries.

The fuzzy approach for smoothening of cluster border is illustrated in Section 4.6.2.

4.6.2 Fuzzy Approach: GDSDC

The output of Phase I, the set of rough clusters, is the input for GDSDC. It initially detects the border cells according to Definition 20. Once the border cells have been found, clustering starts at the pixel level using a fuzzy membership function [Bez81a] as described below.

The border pixels detected for the image shown in Figure 4.3 (a) are shown in Figure 4.3 (c). Suppose, k rough clusters are obtained in the first phase of clustering. Only the pixels in the border cells are checked for possible re-assignment to improve the quality of clusters. Each of the k rough clusters detected has one seed pixel. Let x_j be a pixel in a border cell p of cluster C_i. Since the 8-neighborhood of the cell p may have any other cluster C_j where $j = 1, 2, ...k$, these clusters are the neighbor clusters of cluster C_i and x_j can be assigned to any of the neighboring clusters. The membership of x_j in each of the clusters present in the 8-neighborhood of cell p is calculated using Equation 4.2. x_j is assigned to that cluster for which the membership function has the least value w.r.t. the seed. This process is repeated for all pixels belonging to border cells. The fuzzy membership function [Bez81a], is given by,

$$u_{C_i,x_j} = \frac{1}{\sum_{l=1}^{k} \left[\frac{d(C_i,x_j)}{d(C_l,x_j)}\right]^{\frac{2}{m_f-1}}} \tag{4.2}$$

where x_j is a border pixel, k is the number of clusters detected in the neighborhood of the cell p to which x_j belongs, u is the fuzzy membership matrix such that $u_{ij} \in [0,1]$ is the membership degree of x_j to cluster C_i. $C = \{C_1, C_2, \cdots, C_k\}$ is the set of rough clusters in the neighborhood of cell p. C_i is the current cluster in which the membership of x_j is to be determined and C_l are the clusters present in the 8-neighborhood of cell p, d is a distance measure (Euclidean distance) between a seed of a rough cluster and a border pixel. The factor m_f is called fuzziness and is usually equal to 2 [Bez81a].

The membership for each of the border pixels is computed and assigned to the cluster for which it has the lowest value. It can be easily seen that Figure 4.3 (d) has better quality clusters than those of Figure 4.3 (b).

The steps of the cluster boundary smoothening phase using the fuzzy approach are given below:

Input: q border cells, k seeds corresponding to the k rough clusters obtained from Phase I.

1. Start with an arbitrary border pixel x_j.
2. Find the membership of x_j in each of the k seeds.
3. Assign x_j to the cluster which minimizes the membership function given in Equation 4.2.
4. Repeat steps 1 to 4 till all the border pixels have been considered for reassignment.

4.7 Complexity Analysis

Phase I of both SATCLUS and GDSDC are the same. The complexity analysis for Phase I is given below.

Phase I partitions the dataset and forms of the rough clusters. The partitioning of the dataset into $gr_n \times gr_n$ non-overlapping cells results in a complexity of $O(gr_n \times gr_n)$.

In Phase I, the expansion of the grid cells to form clusters results in $O(cell_p)$ time complexity, where $cell_p$ is the total number of cells in a cluster so formed. $cell_p \ll gr_n \times gr_n$ in the average case. If the number of clusters obtained is k, the overall time complexity for clustering is $O(k \times cell_p)$. Therefore, total time complexity for the rough clustering is $O(gr_n \times gr_n) + O(k \times cell_p)$.

Phase II is different for both SATCLUS and GDSDC. Therefore the computational complexity of both techniques are calculated separately.

In Phase II of SATCLUS, the identification of the q border cells require $O(q)$ times where $q \ll gr_n \times gr_n$. The assignment of r pixels to k clusters using the partitioning approach requires $O(k \times r)$ time, where r is the total number of pixels in q border cells. Therefore, total time complexity for Phase II is $O(q) + O(k \times r)$.

Overall time complexity of SATCLUS is $O(gr_n \times gr_n) + O(k \times cell_p) + O(q) + O(k \times r)$. $O(k \times r)$ dominates the overall time complexity.

In Phase II of GDSDC, the identification of the q border cells require $O(q)$ time where $q \ll gr_n \times gr_n$. The assignment of r pixels to k clusters using the fuzzy membership function requires $O(k \times r)$ time, where r is the total number of pixels in q border cells. Therefore, total time complexity for Phase II is $O(q) + O(k \times r)$.

Overall time complexity of GDSDC is $O(gr_n \times gr_n) + O(k \times cell_p) + O(q) + O(k \times r)$. $O(k \times r)$ dominates the overall time complexity.

Therefore, we see that the time complexity of both algorithms is the same and the user may choose either of the techniques without any time penalty. The added advantage of GDSDC is that it can identify mixed pixels well.

4.8 Performance Evaluation

To evaluate the proposed methods in terms of quality of clustering, we use several synthetic and real datasets. We implement the methods using Java in the Windows environment on a PIV processor with 1 GHz processor speed and 256 MB RAM. The satellite image datasets used have been divided into two parts according to their resolution as discussed below.

4.8.1 Satellite Images with Low Resolution

a) *Dataset I*: This dataset is a Landsat MSS image as shown in Figure 4.5 (a). Landsat Multi Spectral Scanner (MSS) was a sensor on board Landsats 1 through 5 and acquired images of the earth nearly continuously from July 1972 to October 1992, with an 18-day repeat cycle for Landsats 1 through 3 and a 16-day repeat cycle for Landsats 4 and 5. Landsat MSS image data consist of 4 spectral bands although the specific band designations change from Landsats 1-3 to Landsats 4-5. The resolution for all bands is of 79 m, and approximate size is 170 km North-South by 185 km East-West. The clusters obtained from the image of Figure 4.5 (a) are shown in Figure 4.5 (b) for SATCLUS and Figure 4.5 (c) for GDSDC.

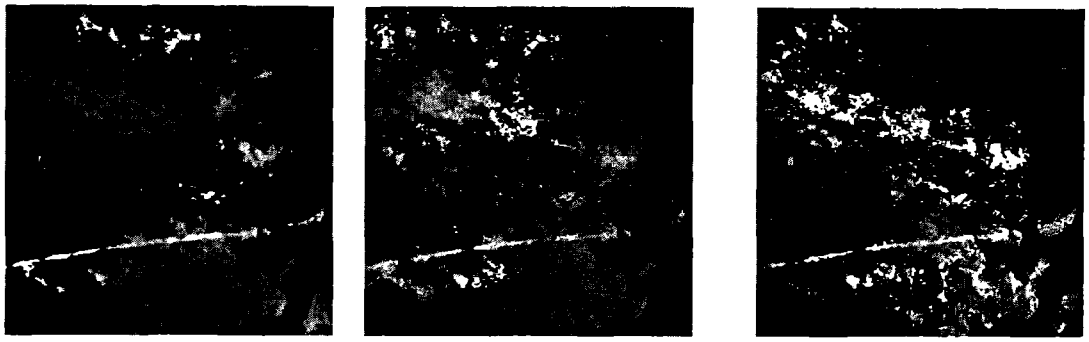


Figure 4.5: a) Landsat-MSS image data, b) Output of SATCLUS c) Output of GDSDC

b) *Dataset II*: This dataset, shown in Figure 4.6 (a) was obtained from the Indian Remote Sensing Satellite which is a circular sun-synchronous satellite. It rotates around the earth at the rate of 14 orbits per day, at an altitude of 904 km and a repeat cycle of 22 days. This satellite has two sensors LISS (Linear Imaging Self Scanner)-I and LISS-II. LISS-I has a spatial resolution of 72.5 m $\times$72.5 m while LISS-II has resolution of 36.25 m $\times$36.25 m. The IRS-1A image of Kolkata was taken by LISS-II sensor in the wavelength range $0.45\mu m$ - $0.86\mu m$. The full spectrum range is decomposed into four spectral bands namely blue band of wavelength $0.45\mu m$-$0.52\mu m$, green band of wavelength $0.52\mu m$ - $0.59\mu m$, red band of wavelength $0.62\mu m$ - $0.68\mu m$ and near-infrared (NIR) band of wavelength $0.77\mu m$ - $0.86\mu m$. This dataset shows an area around Kolkata in the NIR band. There is a prominent black stretch across the image representing the river *Hoogly*. The prominent light patch at the bottom right corner is the *Salt Lake Stadium* and the black patches nearby are the fisheries. Two parallel lines at the upper right hand side of the image correspond to airport runways in the *Dumdum* airport. Other than these there are several water bodies, roads and open spaces in the image.

Both SATCLUS and GDSDC automatically detects four clusters for this data as observed in Figure 4.6 (b) and Figure 4.6 (c). From our back ground knowledge, we can infer that these four clusters correspond to the four classes: Water Bodies (black color), Habitation and City area (deep gray color), Open space (light gray

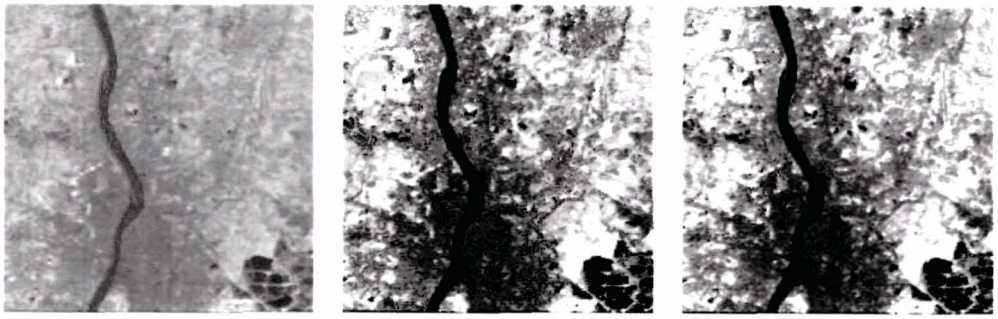


Figure 4.6: a) IRS Kolkata b) Output of SATCLUS c) Output of GDSDC

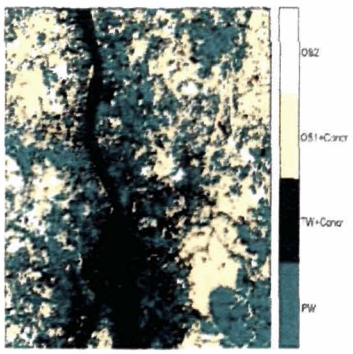


Figure 4.7: FCM clustering of Figure 4.6 a)

color) and Vegetation (white color). The river *Hoogly*, stadium, fisheries, city area as well as the airport runways are distinctly discernible in the output image. The predominance of city area on both sides of the river, particularly at the bottom part of the image is also correctly classified. This area corresponds to the central part of Kolkata city.

Figure 4.7 shows the Kolkata image partitioned using the FCM algorithm. From the figure, we note that the river Hoogly and the city area are not been correctly classified. In fact, these are classified as belonging to the same class. Another misclassification is that the whole Salt Lake City area has been put in one class. Although some portions have been correctly identified such as the canals, the Dumdum airport runways and the fisheries there is still a significant amount of confusion in the FCM clusters.

Figure 4.8: a) Cartosat-1 of Sonari b) Output of SATCLUS c) Output of GDSDC

4.8.2 Satellite Images with High Resolution

The experiments using the images presented next are aimed at handling two different types of terrains (plains and hills) to gauge the variation in classification accuracy. Images of the plain built up area of Sonari in Sibsagar district of Assam, the Borapani area of Meghalaya and a part of Shillong city of Meghalaya are considered for this study.

a) *Dataset III*: This dataset[1] was acquired from the Cartosat-1 remote sensing satellite using the panchromatic (PAN) cameras that take black and white stereoscopic pictures of the earth in the visible region of the electromagnetic spectrum. The swath covered by these high resolution PAN cameras is $30m \times 30m$ and their spatial resolution is 2.5 m and wavelength of $0.5 - 0.85\mu m$. Figure 4.8 (a) shows the Cartosat-1 image of a plain built up area of in Sibsagar district of Assam. Some characteristic regions in the image are the river *Brahmaputra* shown in black, spirally cutting across the middle of the image. We see roads, agricultural land and human settlements as well. The proposed clustering methods automatically detect all the 5 clusters (Figure 4.8) corresponding to river, road, agricultural land, water bodies and human settlements.

b) *Dataset IV and V*: These two datasets[1] show two different views of the Borapani area of the state of Meghalaya obtained from the IRS P6 LISS IV sensor that

[1]These datasets have been obtained from North East Space Application Center, Umium, Meghalaya

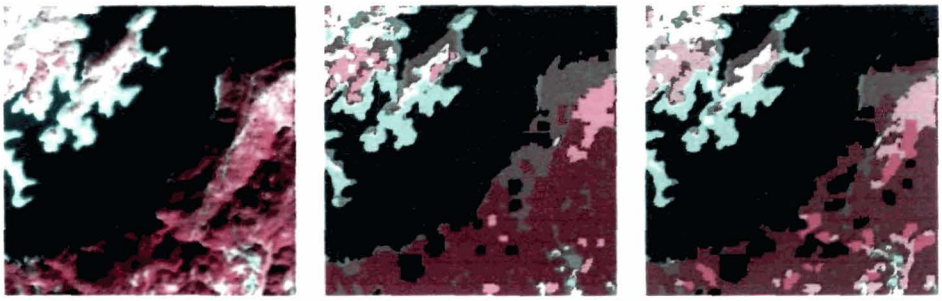


Figure 4.9: a) IRS of Borapani b) Output of SATCLUS c) Output of GDSDC

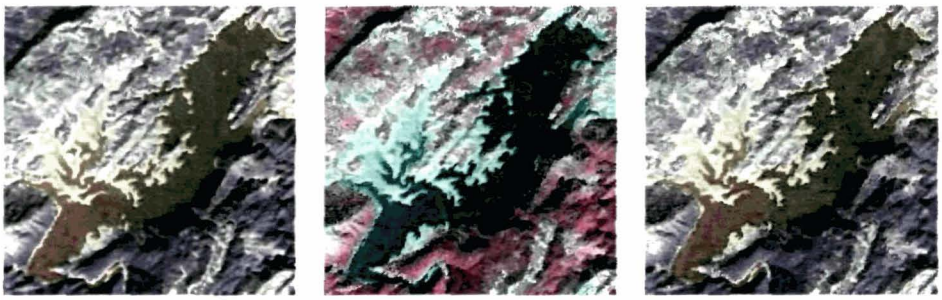


Figure 4.10: a) IRS of Borapani (another view) b) Output of SATCLUS c) Output of GDSDC

has a spatial resolution of 5.8 m. The full spectral range is divided into four spectral bands $0.5 - 0.62\mu m$ (Green), $0.5 - 0.62\mu m$ (Red), $0.5 - 0.62\mu m$ (Infrared) and $0.5 - 0.62\mu m$ (Blue). The characteristic regions in the image shown in Figure 4.9 (a) are Deep water (deep blue color), Wetlands (light blue color), Vegetation (Red and Pink colors) and Open spaces (White color). Executing the SATCLUS and GDSDC algorithms with this image resulted in the detection of the above four classes as shown in Figure 4.9 (b) and (c).

The characteristic regions in Figure 4.10 (a), another image of Borapani, Meghalaya, are the water (dark color), Wetlands (light yellowish-green color), Vegetation (violet color) and Open spaces (light green color). Both SATCLUS and GDSDC clustered the image into five classes as shown in Figure 4.10. showing the clustered regions as deep water (dark blue), wetland (sky blue), vegetation (pink), open spaces (white) and pond water (black). We see that the water body at the left hand top corner of

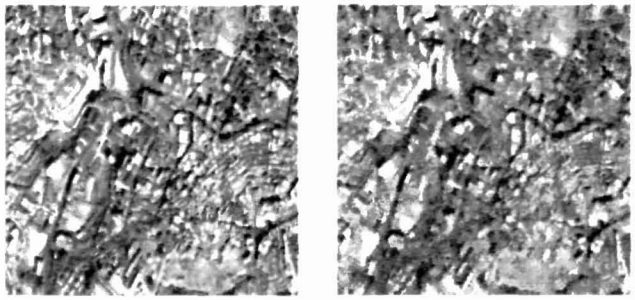


Figure 4.11: a) Ikonos image of Shillong city, Meghalaya, b) Output of SATCLUS c) Output of GDSDC

the image is detected, corresponding well to the ground information available.

c) *Dataset VI*: This dataset[1] was acquired from IKONOS over the area of Shillong city of Meghalaya. IKONOS is a commercial earth observation satellite and offers multi-spectral and panchromatic imagery and has a spatial resolution of 4m and 1 m, respectively. The spectral resolution is divided into four bands namely $0.445 - 0.516\mu m$ (Blue), $0.506 - 0.595\mu m$ (Green), $0.632 - 0.698\mu m$ (Red) and $0.757 - 0.853\mu m$ (NIR). The swath covered is 11 km $\times$ 11 km in a single scene.

Figure 4.11 (a) shows the IKONOS image of Shillong city, Meghalaya. The characteristic regions in this image are the concrete structures, roads and open spaces. The clustered image output obtained by the proposed techniques is shown in Figure 4.11. These relate well with the ground information known to us.

From the experimental results given above, we observe that both the methods, SATCLUS and GDSDC, are highly capable of detecting clusters of all shapes. The set of clusters produced by of the proposed methods are further validated using two cluster validity measures: Cluster homogeneity measure [SMKS03] and the β measure [PGS00].

4.8.3 Cluster Validity

To validate the quality of clustering, we use two validity measures: Homogeneity and β-measure.

a) *Cluster Homogeneity Measure*: Homogeneity measures the quality of clusters on the basis of the definition of a cluster: objects within a cluster are similar while objects in different clusters are dissimilar. The homogeneity measure used is that of overall average homogeneity used in [SMKS03]. It is calculated as follows.

i) Compute the average value of similarity between each object o_i and the seed of the cluster to which it has been assigned.

$$H(C_i) = \frac{1}{||C_i||} \sum_{o_i \in C_i} Similarity(o_i, o_s) \tag{4.3}$$

where o_s is the centroid of C_i.

ii) Calculate the average homogeneity for the set of clusters C according to the size of the clusters as

$$H_{avg} = \frac{1}{||N||} \sum_{C_i \in C} ||C_i|| \dot{H}(C_i) \tag{4.4}$$

Where N is the total number of objects.

Here, o_i refers to a pixel and the centroid of the cluster is represented by the seed of the cluster (o_s). The homogeneity values for the satellite images shown earlier are given in Table 4.1. Homogeneity values are reported for both SATCLUS (using a partitioning approach in Phase II) and GDSDC (using a fuzzy approach in Phase II).

b) *β Cluster Validity Measure*: The set of clusters for the remote sensing images obtained above have also been evaluated quantitatively using the β index as in [PGS00]. Let n_i be the number of pixels in the i^{th} cluster ($i = 1, \cdots, c$). Let X_{ij} be the vector (of size 3×1) of the HSI values of the j^{th} pixel ($j = 1, \cdots, n_i$) for all the images in cluster i, and $\bar{X}_i$ the mean of n_i HSI values of the i^{th} cluster. Then, β is defined as [PGS00]:

$$\beta = \frac{\sum_{i=1}^{c} \sum_{j=1}^{n_i} (X_{ij} - \bar{X})^T (X_{ij} - \bar{X})}{\sum_{i=1}^{c} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^T (X_{ij} - \bar{X}_i)} \tag{4.5}$$

Table 4.1: Homogeneity values for SATCLUS and GDSDC for some satellite image datasets

Dataset	SATCLUS	GDSDC
Dataset I	0.984	0.983
Dataset II	0.961	0.960
Dataset III	0.9908	0.9808
Dataset IV	0.9913	0.9813
Dataset V	0.9815	0.9715
Dataset VI	0.9806	0.9701

where n is the size of the image and $\bar{X}$ is the mean HSI value of the image. Note that X_{ij}, $\bar{X}$, and $\bar{X}_i$ are all 3×1 vectors.

The above measure is the ratio of the total variation and within-cluster variation and is widely used for feature selection and cluster analysis [MMP02]. For a given image and a value for c (number of clusters), the higher the homogeneity within the segmented regions, the higher the β value. That both SATCLUS and GDSDC have higher β values than comparable algorithms is seen in Table 4.2.

Table 4.2: Comparison of beta value and CPU time for different clustering algorithms

Method	k-means [McQ67]	Astrahan's [Ast70]	Mitra's [MMP02]	Acharyya's [AK07]	SATCLUS	GDSDC
β	5.30	7.02	9.88	3.84578	17.82	12.63
CPU time (in hrs)	0.11	0.71	0.75	unknown	0.08	0.09

4.9 Discussion

This chapter has reported two grid-density based clustering methods, SATCLUS and GDSDC for high-resolution satellite image data[2]. The cluster creation using

[2]This work is an outcome of a research project funded by ISRO under the RESPOND scheme

grid cells detects the rough cluster structures. This is because after expansion of a cluster the method searches for the next candidate cell that has a variation in the hue value in the dataset. The process expands the new region till there is again a hue value variation. This process iterates till all the cells have been classified. SATCLUS uses a partitioning based process of smoothening the cluster borders, giving a finer set of clusters since the cluster expansion based on cells may sometimes misclassify the border points. GDSDC exploits a fuzzy membership function for smoothening the cluster borders. This also helps in the handling of mixed pixels in the images. Both techniques reassign a border point to the most relevant cluster. This is because a border point may be misclassified during the cell based expansion. Reassignment improves the quality of the clusters to a great extent. Based on experimental results, both SATCLUS and GDSDC can detect clusters of all shapes. The homogeneity scores for the clusters are also quite good. Based on the CPU time needed and β measure, the methods perform better than several other comparable algorithms ([McQ67], [Ast70], [MMP02]).

In recent years there has been tremendous progress in the data accumulation techniques. This in turn has resulted in the generation of huge amounts of data. Handling such voluminous data is a challenge in the field of data mining. Parallel and distributed techniques help in handling such large amounts of data. In the next chapter, we present two distributed clustering techniques for spatial data.

Chapter 5

Distributed Grid-Density based Clustering

Identifying clusters in large spatial data is a difficult task due to the high amount of processing time needed in handling voluminous data. Distributed and parallel clustering approaches help to reduce the time needed by distributing the processing to different machines.This also improves the response time.

In this chapter, we discuss two distributed clustering techniques that can be applied to handle large scale 2D spatial datasets and large satellite image datasets with high resolution. The first technique (DGDCT) aims to identify embedded clusters in any large 2D spatial datasets with improved clustering quality. The second method is a distributed Grid-Density based Satellite data Clustering technique, DisClus, that can detect clusters of arbitrary shapes and sizes over large, high resolution, multi-spectral satellite datasets. Both techniques are implemented using a client server approach, where the huge dataset stored in the server is partitioned into almost k_p equal partitions that are used by k_p clients to identify the clusters in parallel for each partition. Finally, the clusters obtained from the k_p clients are merged at the server for the final results. Experimental results establish the superiority of the techniques in terms of scale-up, speedup as well as cluster quality, in comparison to similar algorithms.

5.1 Introduction

Extraction of hidden information from huge datasets is a challenging task in data mining. With the increase in the amount of spatial data, the need for efficient and effective spatial data mining techniques is of utmost importance. Though traditional data mining algorithms may be applicable in some spatial datasets, the challenges imposed by the huge amount of spatial data, need to be addressed. The huge size of datasets, its wide distribution over several sites and the computational complexity are the factors contributing towards the development of parallel and distributed algorithms in the data mining domain. Clustering is the process of division of a dataset into subsets or clusters, so that the similarity of points in each partition is as high as possible while points in different partitions are dissimilar. Parallel and distributed spatial data clustering algorithms may help in addressing the problem mentioned before. Distributed clustering is the partitioning of data into groups, in a distributed environment. Although this field is relatively new, yet it has been explored intensively in the last few years, as the need to employ distributed algorithms has grown signicantly. The evolution of the networking and storage equipment fostered the development of very large datasets and it is infeasible to centrally process these datasets in order to analyze them. Distributed clustering is applied when either the data that need to be processed is distributed, or the computation is distributed, or both of them. If none of these two is distributed, then it is centralized clustering. Parallel and distributed computing is expected to relieve current clustering methods from the sequential bottleneck, provide the ability to scale massive datasets and improve the response time. Such algorithms divide the data into partitions, which are processed in parallel. The results from the partitions are then merged.

Although it is common for data to be distributed in a parallel/distributed environment, the distribution is governed solely by performance considerations. Three main architectures can be proposed for building parallel/distributed DBMSs [1].

1. In a shared-memory system, multiple CPUs are attached to an interconnection network and can access a common region of main memory. A shared memory

[1] http://www.cs.ucf.edu/courses/cop4710/spr2006/notes.html

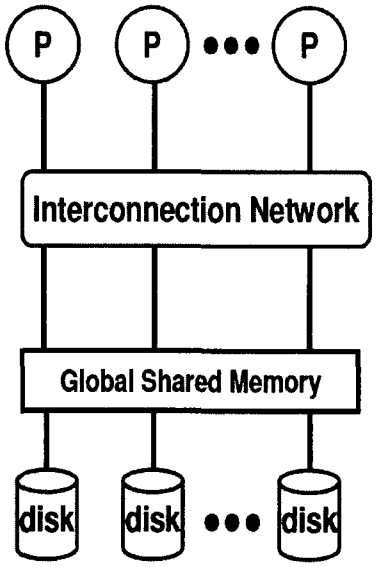


Figure 5.1: The Shared-memory architecture

system is illustrated in Figure 5.1.

2. In a shared-disk system, each CPU has a private memory and direct access to all disks through an interconnection network. Figure 5.2, shows a shared-disk system.

3. In a shared-nothing system as shown in Figure 5.3, each CPU has local main memory and disk space, but no two CPUs can access the same storage area; all communications between CPUs are through a network connection.

Shared memory architecture[2] usually has a block of random access memory that can be accessed by different central processing units (CPUs) in a multiple-processor computer system. This type of architecture is quite easy to program since all processors share a single view of data and the communication between processors can be as fast as memory accesses to the same location. The problem with these systems is that many CPUs need fast access to memory and will likely cache memory, which has two complications: (i) CPU-to-memory connection becomes a bottleneck and shared memory computers cannot scale very well. (ii) Whenever one cache is updated with information by a particular processor, the change needs to be reflected to the other

[2]http://en.wikipedia.org/wiki/Shared_memory

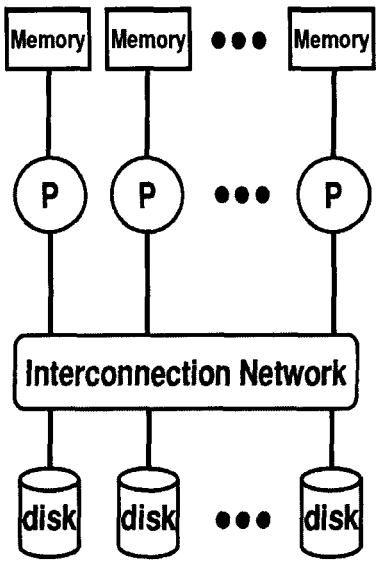


Figure 5.2: The Shared-disk architecture

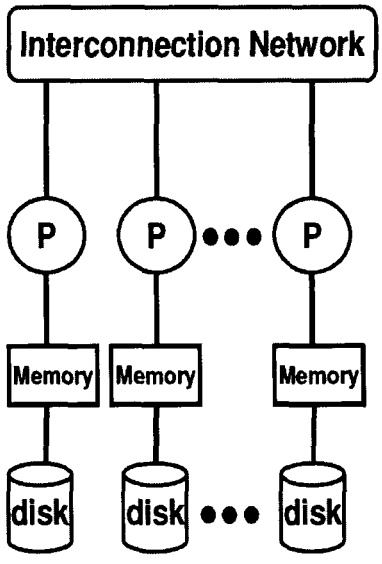


Figure 5.3: The Shared-nothing architecture

processors, otherwise the different processors will be working with incoherent data. If these issues are handled the system works well, provide extremely high-performance access to shared information between multiple processors. On the other hand they can sometimes become overloaded and become a bottleneck to performance

In shared-disk architecture[3] all processors can access the same disks with about the same performance, but are unable to access each other's RAM. With the advent of Network Attached Storage devices (NAS) which allow a storage device on a network, is to be mounted by a set of nodes, shared disk has become increasingly popular. One key advantage of shared-disk systems over shared-nothing is in usability, since DBAs of shared-disk systems do not have to consider partitioning tables across machines. In this system the failure of a single DBMS processing node does not affect the other nodes' ability to access the full database which is not the case with shared-memory systems (that fail as a unit), and shared-nothing systems (that lose at least some data upon a node failure).

Shared nothing architecture (SNA)[4] is a distributed computing architecture consisting of multiple nodes such that each node has its own private memory, disks and input/output devices independent of any other node in the network. Each node is independent and self-sufficient, and shares nothing across the network. Therefore, there are no points of contention across the system and no scope for data sharing or system resources. This type of architecture is highly scalable and has become quite popular for web development because of its scalability. An SN system typically partitions its data among the different nodes such that each node may be responsible for handling a particular task (interacting with different types of users, handling different types of queries, serving different geographic areas etc), or it may require every node to maintain its own copy of the application's data, using some kind of coordination protocol to interact with other nodes as required.

The disadvantage of the shared-memory and shared-disk architectures is interference. As the number of CPUs are increased, existing CPUs are slowed down because

[3]http://www.coders2020.com/explain-the-shared-disk-architecture

[4]http://en.wikipedia.org/wiki/Shared_nothing_architecture

of the increased contention for memory accesses and network bandwidth. The shared-nothing architecture requires more extensive reorganization of the DBMS code, but it has been shown to provide a linear speed-up and linear scale-up. Linear speed-up occurs when the time required by an operation decreases in proportion to the increase in the number of CPUs and disks. Linear scale-up occurs when the performance level is sustained if the number of CPUs and disks are increased in proportion to the amount of data. As a result, ever-more-powerful parallel database systems can be constructed by taking advantage of the rapidly improving performance for single-CPU systems and connecting as many CPUs as desired.

In this chapter, we propose two distributed clustering techniques that use the shared-nothing architecture. The first technique presented in Section 5.3 is capable of identifying arbitrary shaped embedded clusters as well as multi-density clusters over large spatial datasets. The second technique is reported in Section 5.4 and has been applied over huge satellite images to detect the various clusters present in the data.

5.2 Related Work

This section presents a selected survey on some of the distributed and parallel algorithms.

5.2.1 Distributed and Parallel Clustering Techniques

For the past few decades the mainstream data clustering technologies have been fundamentally based on centralized operation; datasets were of small manageable sizes, and usually resided on one site that belonged to one organization. Today, data is of enormous sizes and is usually located on distributed sites; the primary example being the Web. This created a need for performing clustering in distributed environments. Distributed clustering solves two problems: infeasibility of collecting data at a central site, due to either technical and/or privacy limitations, and intractability of traditional clustering algorithms on huge datasets.

In [BBD04], a parallel implementation of the DBSCAN algorithm based on low cost distributed memory multi-computers is presented. Here, a centrally located dataset is spatially divided into nearly equal partitions with minimum overlap. Each such partition is sent to one of the processors for parallel clustering. The clustering results of the partitions are then collected by the central processor in an orderly manner and they are merged together to obtain the final clustering. The algorithm is scalable both in terms of speedup and scale-up and significantly reduces the computation time.

In [DM99], a parallel version of the k-means algorithm was proposed based on shared nothing architecture. This algorithm was designed based on the Single Program Multiple Data (SPMD) model having several processors, each having its own local memory, connected together with a communication network. Each processor, or node, receives only a segment of the data that needs to be clustered. One of the nodes selects the initial cluster centroids, before sending them to the others. New distances between centroids and data points are computed independently, but after each iteration of the algorithm, the independent results must be aggregated or reduced. This is done using the MPI (Message Passing Interface). The reduced centroids obtained after the last iteration represent the final result of the clustering process.

Another parallel version of DBSCAN, called PDBSCAN [XJK99], also uses a shared-nothing architecture with multiple computers interconnected through a network. Here, as a data structure, the dR*-tree was introduced which is a distributed spatial index structure in which the data is spread among multiple computers and the indexes of the data are replicated on every computer. The master distributes the entire dataset to every slave. Each slave locally clusters the replicated data and the interference between computers is minimized due to local access of data. The slave-to-slave and master-to-slaves communication is done via message passing. The master manages the task of dynamic load balancing and merges the result produced by the slaves. PDBSCAN offers nearly linear speedup and has excellent scale-up and size-up behavior.

In [JKP03], a Density Based Distributed Clustering(DBDC) algorithm was presented which can be used in the case when the data to be clustered is distributed and infeasible to centralize. DBDC works by first clustering the data locally at different sites independent of each other. The aggregated information about locally created clusters are extracted and transmitted to a central site. On the central site, a global clustering is performed based on the local representatives and the result sent back to the local sites. The local sites update their clustering based on the global model, that is, merge two local clusters to one or assign local noise to global clusters. For both the local and global clustering, density-based algorithms are used. This approach is scalable to large datasets and gives clusters of good quality.

In [FLPT00], a parallel version of the AutoClass system, P-AutoClass is described. In [JK99], a Collective Hierarchical Clustering (CHC) algorithm is reported for analyzing data that is heterogeneously distributed, with each site having only a subset of all features. First, a local hierarchical clustering is performed on each site. Afterwards, the obtained dendrograms are sent to a facilitator which computes the global model, using statistical bounds. The aggregated results are similar to centralized clustering results, making CHC an exact algorithm.

The algorithm P2P K-means, developed by Datta et. al. [DGK06] is one of the first algorithms developed for P2P systems. Each node requires synchronization only with the nodes that it is directly connected to, or its neighborhood. Only one node initializes the centroids used for k-means, which are then spread to the entire network. The centroids are updated iteratively. Before computing them at step i, a node must receive the centroids obtained at step $i-1$ by all of its neighbors. When the new centroids of a particular node do not suffer major modications, then the node enters a terminated state, where it doesn't request any centroids, but it can response to requests by neighbors. Node or edge failures and additions are also accounted for by P2P k-means, making it suitable for dynamic networks. P2P K-Means algorithm was proposed in [BGM+06] for distributed clustering of data streams in a peer-to-peer sensor network environment.

Jin R. et al. [JGA06] presented a distributed version of Fast and Exact K-Means (FEKM) algorithm, which collected sample data from each data source, and communicated it to the central node. The main data structure of FEKM i.e. the cluster abstract table is computed and sent to all data sources to get global clusters.

In [ALKK07], the authors proposed a lightweight distributed clustering technique based on a merging of independent local sub clusters according to an increasing variance constraint. The key idea of this algorithm is to choose a relatively high number of clusters locally, or an optimal local number using an approximation technique, and to merge them at the global level according to an increasing variance criterion which requires a very limited communication overhead.

Le-Khac N. et al. [LKAK07] presented an approach for distributed density-based clustering. The local models are created by DBSCAN at each node of the system and these local models are aggregated by using tree based topologies to construct global models.

In [TMEDF08], the authors introduced a method to define intuitionistic fuzzy partitions from the result of different fuzzy clustering algorithms such as FCM, entropy based FCM and FCM with tolerance. In this approach, the intuitionistic fuzzy partition permits to cope with the uncertainty present in the execution of different fuzzy clustering algorithms with the same data and with the same parameterization.

In [DGK09], Datta et. al. propose two approximate K-means clustering algorithms that work on uniformly sampled peers. The first algorithm is designed to operate in a dynamic P2P network that can produce clusterings by local synchronization only. The algorithm has been observed empirically to produce accurate clustering results with respect to centralized K-means clustering. However, it cannot offer an analytical accuracy guarantee. Therefore, the second algorithm is proposed which works by taking a uniform random sample of nodes from a static P2P network. This algorithm provides an analytical accuracy guarantee.

Another intuitionistic fuzzy based distributed clustering algorithm is presented in [VTP10] for homogeneously distributed datasets. The process is carried out in two different levels: local level and global level. In local level, numerical datasets are converted into intuitionistic fuzzy data. Modified fuzzy C-Means algorithm is then used to cluster this data independently from each other. In global level, global centroid is computed by clustering all local cluster centroids. The global centroid is again transmitted to local sites to update the local cluster model.

5.2.2 Discussion

Based on our selected survey and experimental analysis, it has been observed that density based approach is most suitable for quality cluster detection over massive datasets. Almost all clustering algorithms require input parameters, determination of which are very difficult, especially for real world datasets containing high dimensional objects. Moreover, the algorithms are highly sensitive to those parameters. Distribution of most of the real-life datasets are skewed in nature, so, handling of such datasets for all types for qualitative cluster detection based on a global input parameter seems to be impractical. Also handling high dimensional data is a challenging task. The performance of most of the algorithms aimed to identify quality clusters for 2D spatial data degrades with the increase in dimensionality. Algorithms like DBSCAN [EKSX96] and GDBSCAN [SEKX98], which give good quality clusterings, do not work for high dimensional data. Often, the algorithms present in the literature can be found to identify clusters over large spatial data at an abstract level, however, some applications demand for identification of these at a more detailed or finer level. None of the techniques discussed above, is capable to handle the embedded or intrinsic cluster detection problem over massive datasets successfully. An algorithm which is capable of handling voluminous data and at the same time effectively detects nested or embedded clusters in presence of noise is of utmost importance. The grid density based clustering algorithm(GDCT) discussed in Chapter 3 finds clusters according to the structure of the embedding space. For handling massive datasets, a distributed clustering technique based on GDCT is presented which can effectively address the scalability problem. Better speedup and scale-up are the major attractions of the proposed technique.

5.3 Distributed Grid-Density based Clustering Technique (DGDCT)

This section presents a Distributed Grid-Density based Clustering Technique (DGDCT) capable of identifying arbitrary shaped embedded clusters as well as multi-density clusters over large spatial datasets. For handling massive datasets, we implemented our method using a shared-nothing architecture where multiple computers are interconnected over a network. We consider a system having k_p-nodes where the entire dataset D is located in any of the nodes (say initiator node). DGDCT can be initiated in any of the available nodes (computers). The initiator node starts a partitioning strategy thereby dividing the whole dataset into partitions and then distributing the partitions to each of the available computers on the network (one partition is also retained by itself). The initiator node executes a fast partitioning technique to generate the k_p initial partitions. The partitions are then sent to k_p nodes (including itself) for cluster detection using a grid-density based clustering technique (GDCT) which can operate over variable density space. Every node clusters only its local data. The initiator node manages the task of dynamic load balancing. Finally, the local cluster results are received from the nodes at the initiator node and a merger module is invoked to obtain the final cluster results. Basically the technique works in three phases and the output of the previous Phase becomes the input of the current Phase. Next, we describe the architecture as shown in Figure 5.4, phase-wise.

The proposed DGDCT can be found significant in view of the following issues:

1. Embedded cluster Detection,
2. Handling of single linkage problem,
3. Handling of huge datasets (Scalability),

The first two advantages is due to the fact that the clustering algorithm as given in Section 3.4 can identify embedded clusters and can handle the problem of single linkage which is inherent to most of the density based algorithms. DGDCT is scalable to huge datasets as it uses a fast partitioning technique to distribute the huge

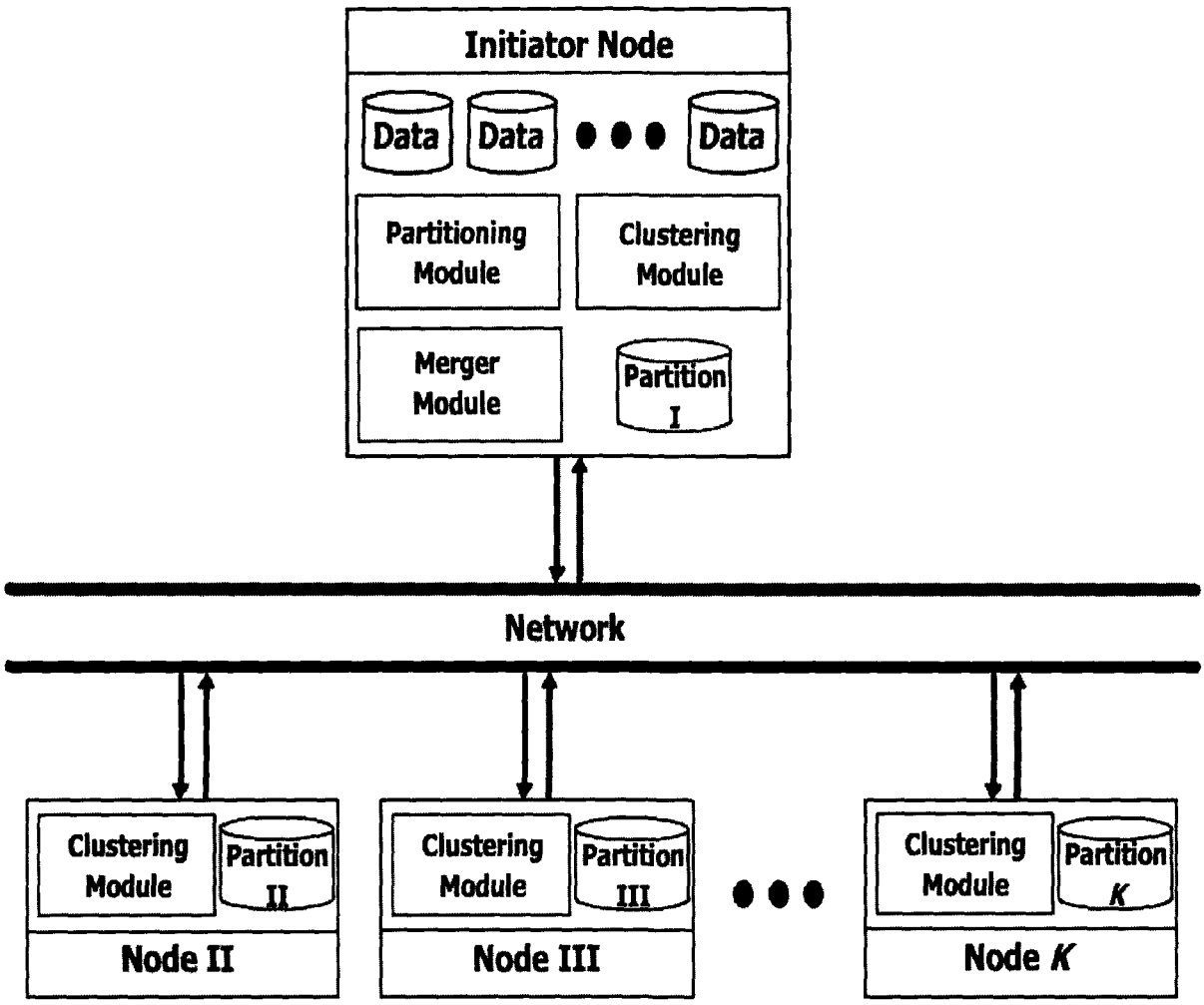


Figure 5.4: The architecture of the Proposed Technique

data to different nodes for local clustering and finally merges the cluster results at the initiator. The actual clustering is done on the distributed data and hence the processing load reduces even in case of huge datasets. This has been explained in detail in Section 5.3.1.

An overview of the hardware architecture is shown in Figure 5.4. It consists of a number of nodes (e.g. PCs) connected via a network (e.g. Ethernet).

5.3.1 Phase I: Partitioning the dataset

Phase I of the architecture is executed in one of the nodes (initiator node). The dataset is spatially divided into equal sized square grid cells and density of each grid cell is computed. The square mesh is then partitioned with some overlap between adjacent partitions and distributed over k_p available computers (nodes). No subsequent movement of data between partitions will take place.

Initially, the data space is divided into $gr_n \times gr_n$ non-overlapping square grid cells, where gr_n is a user input, and maps the data points to each cell. It then calculates the density of each cell. Assuming, the grid mesh D^* contains the set of $gr_n \times gr_n$ objects say, $D^* = O_0, O_1, O_2, \cdots, O_{(gr_n \times gr_n)-1}$. Suppose, $O_j = (a_{0j}, a_{1j}, a_{2j}, \cdots, a_{(n-1)j}; d_n)$ represents a grid cell with n real-valued attributes a_i, $i = 0, \cdots, n-1$ and density d_n. The i^{th} attribute value of object O_j is drawn from domain a_j. If there are k_p clients, the grid mesh D^* is partitioned into k_p subsets $D_0, D_1, \cdots, D_{k_p-1}$ ordered in sequence. We refer the clients by the corresponding partition D_j that it receives for processing.

$$D^* = D_0 \cup D_1 \cup D_2 \cup \cdots \cup D_{k_p-1}$$

$$D_i \cap D_j = \phi; i, j = 0, 1, \cdots, k_p - 1$$

The partially overlapped partitions are shown in Figure 5.5 for 2D case. An overlap of one grid cell occurs between two adjacent partitions. The overlapped regions are much smaller than the partitions. The grid cells in the overlapped regions are locally clustered in both the adjacent partitions. Thus they provide the information for merging together the local clustering results of two adjacent partitions. The overlapped width should be at least one cell width because adjacent cells are neighbors according to Definition 3. The grid mesh D^* is partitioned in this manner based on the values of a selected attribute of the data objects say a_s as in [BBD04]. The values of a_s have a range of $[min\_a_s, max\_a_s]$. We need to select $(k_p + 1)$ constants in the given range. Let $c_i^s, i = 1, \cdots, k_p + 1$ represents the constants such that $c_1^s = min\_a_s$, $c_{k_p+1}^s = max\_a_s$ and $c_i^s < c_{i+1}^s$. Therefore the overlapped region can be represented as:

$$D_i = \exists j (O_j \in D^*) \mid c_i^s - cell\_width \leq a_{sj} \leq c_{i+1}^s, i = 2, \cdots, k_p - 1$$

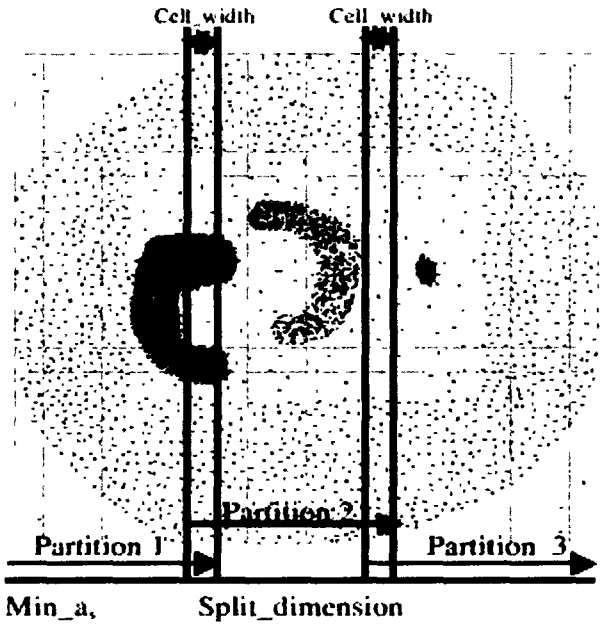


Figure 5.5: Overlapped spatial partitioning of a 2D dataset

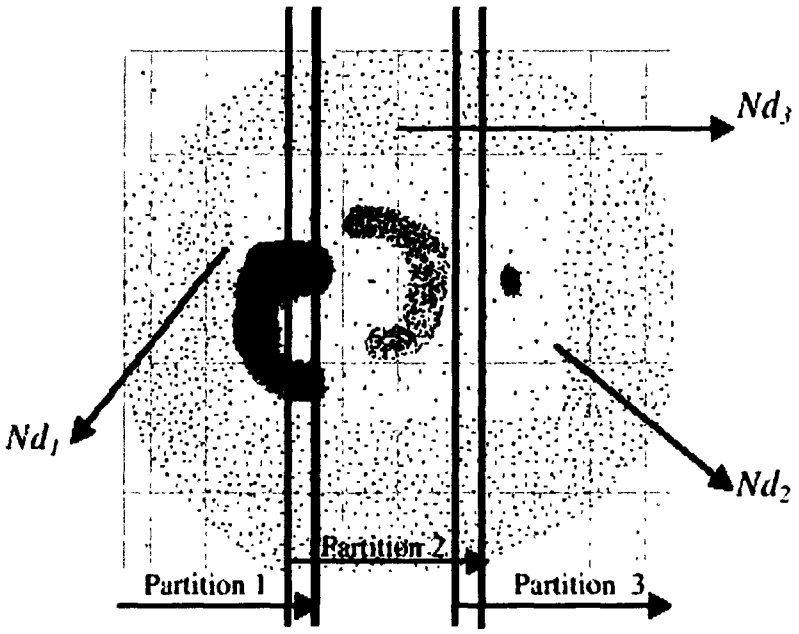


Figure 5.6: Here the dataset is divided into three partitions and transmitted to three computers (Nd_{k_p}) for local clustering, $k_p = 1, 2, 3$

$$D_i = \exists(O_j \in D^*) \mid c_i^s \leq a_{sj} \leq c_{i+1}^s + cell\_width, i = 1$$

$$D_i = \exists j(O_j \in D^*) \mid c_i^s - cell\_width \leq a_{sj} \leq c_{i+1}^s, i = 1$$

The constant c_i^s should be selected in such a manner that $\mid D_j \mid$ becomes nearly equal to $\lceil N/k_p \rceil$, where N is total number of data points in the dataset. Moreover, those grid cells which fall within the overlapped regions are marked. Care has been taken for load balancing. The k_p partitions thus obtained are then sent to k_p nodes for global as well as intrinsic cluster detection (Figure 5.6).

A detailed discussion on the basic sequential algorithm i.e. GDCT is already reported in Section 3.4, however, it was not scalable to huge datasets.

Load Balancing

Partition D_i is sent to processor $P_i, i = 1, 2, \cdots, k_p$ for concurrent clustering. Since no data movement takes place after the partitions are received by the respective nodes, care should be taken so that each processor receives nearly equal number of data objects for processing. This will ensure that all the processors finish the clustering job at the same time provided the processors have same processing speed. If the processing speeds are different, then the input data should be distributed to the processors proportionate to their processing speed. We assume that the speeds of the processors are nearly equal and they receive nearly equal amount of data. For doing this, the range of a_s is divided into intervals of width of one cell-width and the frequencies of data in each interval is counted. Let $b = \lceil (max\_a_s - min\_a_s)/cell\_width \rceil$, $N' = \lceil N/k_p \rceil$, $d_1 = min\_a_s$,

$$d_i = d_{i-1} + cell\_width, i = 2, 3, \cdots, b$$

$$F_i = \exists j(O_j \in D^*) \mid d_i \leq a_{sj} \leq d_{i+1}, i = 2, 3, \cdots, b$$

$$f_i = \mid F_i \mid$$

Now, the constants, c_i^s defined earlier, are computed as $c_i^s = d_s$ such that $\sum_{j=1}^{s} f_j \leq N' \leq \sum_{j=1}^{s+1} f_j$, $i = 1, 2, \cdots, k_p$, which will ensure that each partition gets number of objects nearly equal to N/k_p.

Minimized communication cost

The proposed method saves transmission cost by avoiding inter-node communication during the process of local clustering. To achieve this goal, each concurrent process of GDCT in each of the nodes, $N_d = 1, 2, \cdots, k_p$, should avoid accessing those data located on any of the other computers, because the access of the remote data requires some form of communication. Therefore, nearby objects should be available on the same computer. This is why an overlap of one cell_width has been taken into consideration.

5.3.2 Phase II: Local Clustering

Phase II of the architecture is executed in each of the k_p nodes. This phase plays the actual role of clustering. In this phase, each node executes the GDCT algorithm over the partition of data received from the initiator node to detect the global and nested clusters.

For the partition D_i in node i, the grid cells in it will be assigned cluster_id according to the clusters formed in that partition. The cluster_ids will be used during the server based merging process by the initiator node.

The cluster expansion based on grid cells helps to achieve a significant cost reduction as all the data points are not considered for cluster expansion only the density information of each cell is used. Also, as the cluster_id information are used during Phase III merging process, it saves the cost of merging to a great extent.

5.3.3 Phase III: Merging

In Phase III, the cluster results received from the k_p nodes undergo a simplified, yet faster merging procedure to obtain the final clusters. Since the Phase II process in a node may yield more than one cluster along with the embedded clusters, so there are always possibilities for merging during Phase III operation. The Merger module works as follows:

1. Join the partitions received from the k_p nodes according to their overlapping marks.

2. Consider the marked grid cells (overlapping cells) of the candidate clusters.

   2.1 If any of the marked grid cells is identified by different cluster_ids by different partitions (say l, m), then assign any one of the ids (say l) to that cell.

3. Assign all those cells having the same cluster_id as the replaced id (m) with l.

5.3.4 Complexity Analysis

Phase I: The partitioning of the dataset into $gr_n \times gr_n$ non-overlapping cells results in a complexity of $O(N)$ where N is the total number of data points. The grid mesh D^* is spatially partitioned into k_p partitions with overlap of one cell width which results in a complexity of $O(gr_n \times gr_n)$, where $gr_n << N$. Each of these k_p partitions will have nearly equal (approximately N/k_p) data points. The data points along with the grid information for each of k_p partitions will be sent to the k_p nodes. Therefore $(N/k_p) + t$ points will be sent, where t is the average number of points present in an overlapped region. Next, to transmit these $(N/k_p) + t$ points to each node requires a communication time of $O((N/k_p) + t)$.

Phase II: This phase is executed in each of the k_p nodes. Computing density of the cells in each node requires $O((gr_n \times r) \times ((N/k_p) + t))$, where r is the average number of cells along the selected attribute based on which partitioning in Phase I has been performed. The sorting of cells according to their density results in a complexity of $O((gr_n \times r)log(gr_n \times r))$.

The expansion of the coarse cluster results in $O(m_c)$ time complexity, where m_c is the number of cells in an coarse cluster formed and $m_c << (gr_n \times r)/k_p$ in the average case. Cell subdivision into triangles takes place only in case of the border cells of the coarse cluster and its neighboring cells, Say, there are p border and q neighbor cells where $q >> p$. This step results in a complexity of $O(p + q)$. If the

number of clusters obtained is k then the overall time complexity for the clustering will be $O(k \times m_c \times (p+q))$.

Therefore, total time complexity will be $O((gr_n \times r) \times ((N/k_p) + t)) + O((gr_n \times r)log(gr_n \times r)) + O(k \times m_c \times (p+q))$. Thus the complexity due to density calculation almost dominates the other components, since $(N/k_p)+t) >> (gr_n \times r)$. The clusters detected in this phase are transmitted back to the initiator node with a transmission cost of $O((N/k_p) + t))$.

Phase III: Merging of the clusters obtained from the k_p nodes will take $O(N + k_p.t)$ time.

Thus, the overall time complexity of distributed GDCT will be $O(N) + O(gr_n \times gr_n) + O((N/k_p) + t)) + O((gr_n \times r) \times ((N/k_p) + t)) + O((N/k_p) + t)) + O(N + k_p.t)$. Therefore, the time complexity of DGDCT becomes $O(N)$ since $N >> (gr_n \times gr_n)$.

5.3.5 Performance Evaluation

This section reports an empirical study of DGDCT by measuring execution time, speedup, efficiency and scale-up factors. Since there is no inter-processor communication except for a single processor communicating with each of the remaining processors. Each processor has the same specification i.e. PIV with 1 GHz speed and 128 MB RAM and the processors are connected through Ethernet LAN of speed 10/100 Mbps. measurements. Our implementation is in C in Linux environment and we considered several synthetic datasets containing arbitrary number of arbitrary shaped clusters having 2×10^5, 4×10^5, 6×10^5 and 9×10^5 objects respectively and experimentation was carried out.

The graph of *Parallel Execution Time* is shown in Figure 5.7. From the graph we conclude that the execution time decreases significantly as the number of processors increases.

The *Relative Speedup* curves for two datasets with points $N = 9 \times 10^5$ and 6×10^5 is

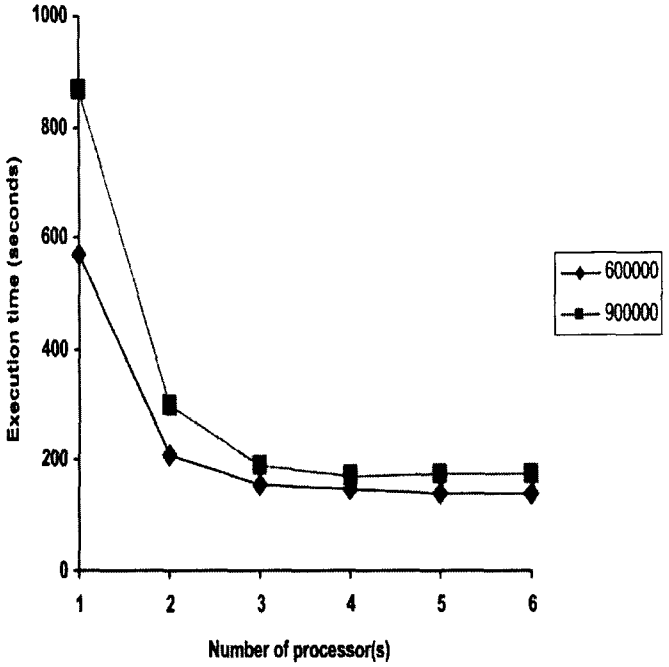


Figure 5.7: Parallel execution time

given in Figure 5.8. The number of dimensions and the number of clusters are fixed for both the datasets.

The scale-up characteristic of the DGDCT has been found to be satisfactory with the increase in the number of processors as can be seen from Figure 5.9. Here, the number of data points is scaled by the number of processors while dimensions and number of clusters are held constant. It is seen from the Figure 5.10 that if two many processors are used then performance degrades.

DGDCT is an effective technique for handling huge 2D numeric datasets qualitatively. However, DGDCT can be applied only to 2D spatial data. For higher dimensional spatial data, DGDCT may be modified in the line of [AGGR98]. For clustering high resolution massive satellite data, a grid density based clustering technique based on DGDCT is reported in the next section.

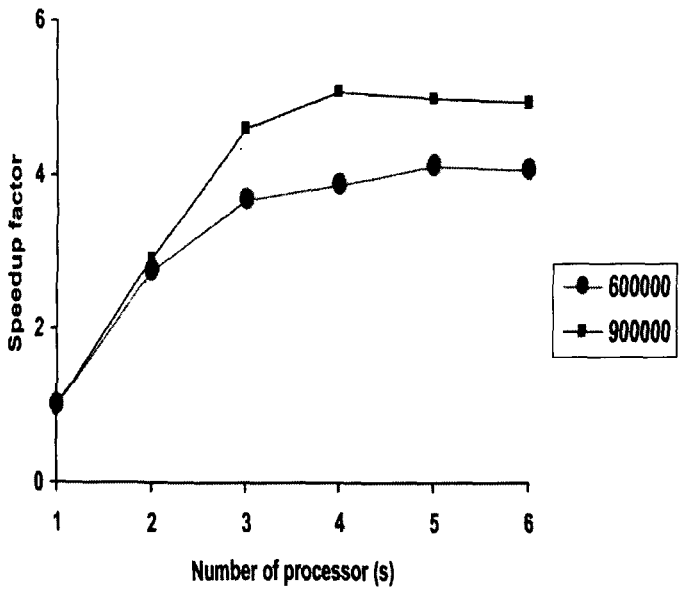


Figure 5.8: Relation between Speedup and number of processors for two datasets.

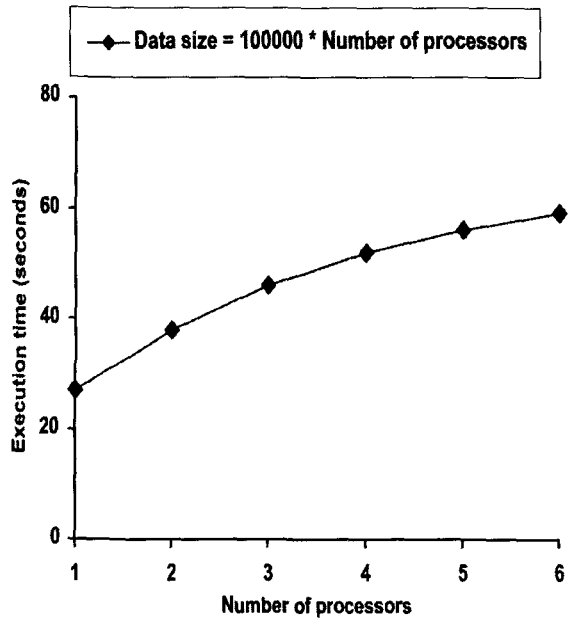


Figure 5.9: Scale-up curve.

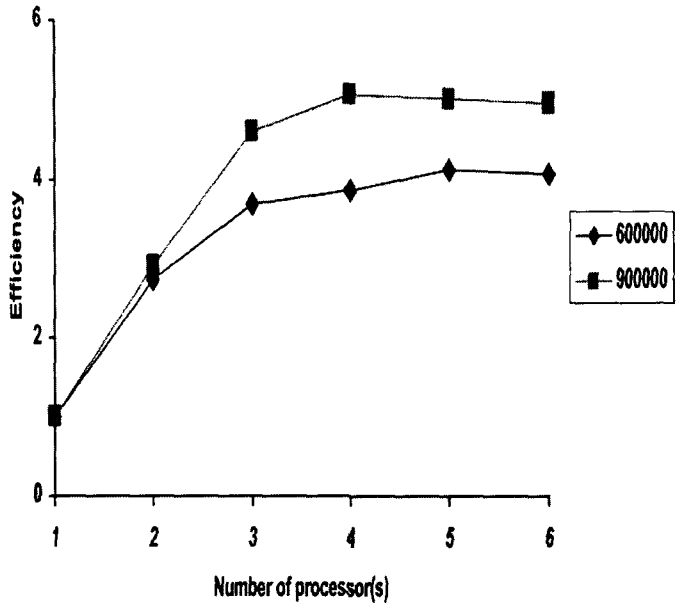


Figure 5.10: Efficiency vs. number of processors employed

5.4 Distributed Grid-Density based Clustering Technique for Satellite Data (DisClus)

Clustering conserves the homogeneous property within a cluster i.e., data points within a cluster are more similar than the data points belonging to different clusters [HK06]. A high resolution satellite image is a remotely sensed image of the earth's surface which is a collection of huge amount of information in terms of number of pixels where each pixel in the image represents an area on the earth's surface. Multi-spectral images are the main type of images acquired by remote sensing. This technology was originally developed for space-based imaging which can capture light of frequencies beyond the visible range of light, such as infrared, which helps to extract additional information that the human eye fails to capture with its receptors for red, green and blue. A multi-spectral satellite image is a digital image comprising of multiple bands where each band represents a particular wavelength of light. Remotely sensed satellite images mainly consists of objects (regions) such as vegetation, water bodies, concrete structures, open spaces, habitation, clouds etc. which are separated due to their different reflectance characteristics, leading to wide variety

of clusters of different sizes, shapes and densities.

Based on our selected survey and experimental analysis, it has been observed that handling large scale data is a challenging task. To discover clusters of varying shapes and sizes effectively over massive spatial datasets is a difficult task. To address these challenges, this chapter presents a distributed grid-density based clustering algorithm (DisClus[5]) based on SATCLUS and GDSDC [SB10] which can detect clusters over high resolution satellite datasets qualitatively. Further, the post processing phase helps in smoothening the bordering regions of clusters. The method was tested and evaluated over satellite datasets and the results has been found satisfactory.

5.4.1 The Proposed DisClus

Like, DGDCT, DisClus also works in three phases. In the first phase, the satellite image is partitioned into regions with marked overlappings at an initiator node and sent to each of the nodes available for clustering. The second phase is executed in each of the participating nodes. In this phase, the clustering of the data for each partition is performed using either one of the techniques, SATCLUS or GDSDC, at each node. Finally, during the third phase, the nodes transmit the cluster results back to the initiator node where the result are merged to get the final result.

The proposed architecture adopts a shared nothing architecture. It considers a system having k_p-nodes where the whole image D is located in any of the nodes (say *node* 1, also referred here as *initiator node*). It executes a fast partitioning technique to generate the k_p initial overlapped partitions. The partitions are then distributed among $k_p - 1$ nodes and one partition is kept at the initiator for cluster detection. Finally, the local cluster results are received from the nodes at this node (*node* 1) and a merger module is used to obtain the final cluster results. Next, each of these phases is explained in brief.

Phase I: In the *initiator node*, the dataset is spatially divided into $gr_n \times gr_n$ non-overlapping square grid cells, where gr_n is a user input, and maps the data points

[5]This work is an outcome of a research project funded by ISRO under RESPOND scheme

to each cell. It then calculates the density of each cell. The grid mesh is then partitioned with some overlap between adjacent partitions and distributed over k_p available computers (nodes). No subsequent movement of data between partitions will take place. An overlap of single grid cell width occurs between two adjacent partitions. The grid cells in the overlapped regions are locally clustered in both the adjacent partitions. Thus, they provide the information for merging together the local clustering results of two adjacent partitions.

Load Balancing: Partition D_i is sent to processor P_i, i=1,$\cdots$, k for concurrent clustering. Since no data movement takes place after the partitions are created and transmitted to the respective nodes till the clustering results are locally available at each node, care has been taken so that each processor receives nearly equal number of data objects (i.e. pixels) for processing. Like DGDCT, here also it is assumed that the speed of all the processors are equal. The range of a_s is divided into intervals of width of one *cell_width* and the frequencies of data in each interval is counted. The load balancing is done in a manner similar to [BBD04] which ensures that each partition gets number of objects nearly equal to N/k_p.

Phase II: In this phase, either SATCLUS or GDSDC (discussed in previous chapter) is executed in each of the k_p nodes over the partition of data received from the *initiator node*. For the partition D_i in node i, the grid cells in it will be assigned *cluster_id* according to the clusters formed in that partition.

The cluster expansion based on grid cells reduces the computation time as data points are not considered for cluster expansion, only the density information of each cell is used. Moreover, the information of the marked cells used during merging process of Phase III saves the cost of merging to a great extent. Finally, Phase II transmits the cluster objects to the *initiator node* along with the *cluster_ids*.

Phase III: Here, the cluster results are gathered from the k_p nodes into the initiator node. A merger module is used which uses the cluster_id information obtained from the partitions to finalize the cluster results. The Merger module first joins the

partitions received from the k_p nodes according to their overlapping marked cells. It considers the marked grid cells (overlapping cells) of the candidate partitions. If any of the marked grid cells is identified by different *cluster_id*s by different partitions (say l, m), then the smallest of the *cluster_id*s (say l) is assigned to that cell. Finally, all those cells having the same *cluster_id* as that of the replaced *cluster_id* (m) is assigned with *cluster_id* l.

The following lemma provides the theoretical basis for the merging process.

Lemma 6. Let m be a marked cell in the overlapping region of two adjacent partitions p_i and p_{i+1} and C_i and C_j are two clusters belonging to p_i and p_{i+1} respectively. If $m \in C_i$ and also $m \in C_j$, then C_i and C_j are merged.

Proof. Suppose, m be a marked cell and cell $x \in C_i$ in p_i and cell $y \in C_j$ in p_{i+1}. If $m \in C_i$ and also $m \in C_j$, then x and y are reachable from m and $m \in C_i \cap C_j$. So, x is connected to y and cells x and y should be in the same cluster. Therefore, clusters C_i and C_j should be merged. □

5.4.2 Complexity Analysis

Since the proposed technique is executed in three phases and each phase is independent of each other, therefore, the total complexity will be the sum of the complexities due to these three phases.

The first phase divides the dataset of N points into $gr_n \times gr_n$ cells which are partitioned into k_p overlapped partitions with a total of $((k_p - 1) \times gr_n)$ overlapped cells. Therefore, this phase results in a complexity of $O(gr_n \times gr_n)$ approximately, where $gr_n << N$. After partitioning, $(N + (k_p - 1) \times t)$ points will be transmitted to k_p nodes, where t is the average number of points present in an overlapped region, results in a complexity of $O((N + (k_p - 1) \times t)$.

The second phase results in a complexity of $O(((gr_n \times gr_n)/k_p + gr_n) + (C^l \times b))$ [SB10], where C^l is the number of clusters detected locally and b is the number of border points obtained in a partition in a node. The clustered points are re-transmitted

to the initiator node with a transmission cost of O($(N + (k_p - 1) \times t)$.

The third phase is responsible for merging of the clusters resulting in atmost O($N + k_p \times t$) time.

Thus, the overall time complexity of DisClus will be O($gr_n \times gr_n$) + O($N + (k_p - 1) \times t$) + O($((gr_n \times gr_n)/k_p + gr_n) + (C^l \times b)$) + O($N + (k_p - 1) \times t$) + O($N + k_p \times t$). Therefore, the time complexity becomes $O(N)$, since $N >> (gr_n \times gr_n)$ and also $N >> ((k_p - 1) \times t)$.

5.4.3 Performance Evaluation

In this section we evaluate the performance of DisClus in light of several real-life satellite image data.

Environment Used

The algorithm was implemented using Java in Windows environment with Pentium IV processor with 1 GHz speed and 256 MB RAM. To smooth out any variation, each experiment was carried out for several times and the average result was taken.

Datasets Used

The algorithm was tested over several real-life satellite images as shown in Table 5.1. The Dataset 1 is shown in Figure 5.11. The clusters obtained from the image of Figure 5.11 are shown in Figure 5.12. Figure 5.13 shows Dataset 2. There is a prominent black stretch across the image which is the river *Hoogly*. The prominent light patch at the bottom right corner is the *Salt Lake stadium* and the black patches nearby are the fisheries. Two parallel lines at the upper right hand side of the image correspond to the airport runway in the *Dumdum* airport. Other than these there are several water bodies, roads, open spaces, etc. in the image.

DisClus automatically detects four clusters for this data as observed in Figure 5.14. From our ground knowledge, we can infer that these four clusters correspond to the

Table 5.1: Results of the clustering algorithm over several multi-spectral satellite images

Serial No.	Dataset	Spectral Bands	Resolution	Clusters Detected
Dataset 1	Landsat MSS	4	79 m	4 clusters
Dataset 2	IRS LISS II image of Kolkata, West Bengal	4	36.25 m	4 clusters
Dataset 3	Cartosat-I image of Sonari, Assam Sonari, Assam	4	2.5 m	5 clusters
Dataset 4	IRS P6 LISS IV image of Borapani, Meghalaya	4	5.8 m	5 clusters

Figure 5.11: Landsat-MSS

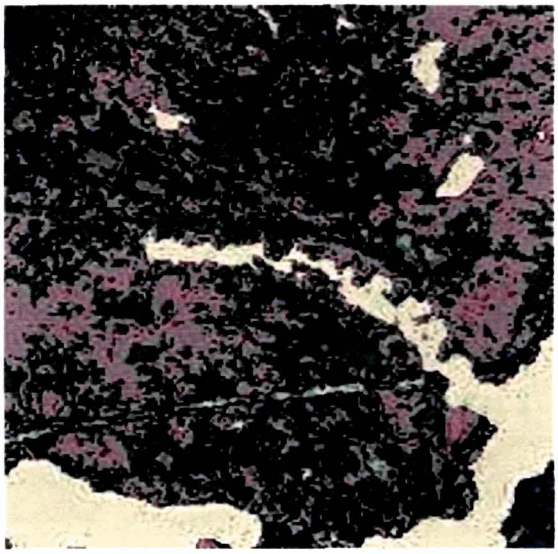


Figure 5.12: DisClus output of Figure 5.11

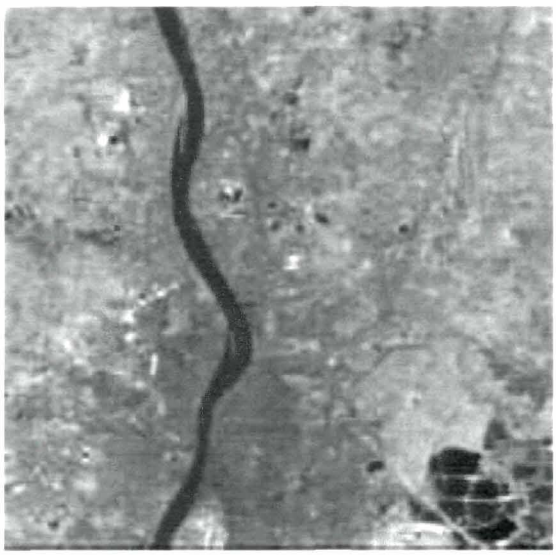


Figure 5.13: IRS Kolkata

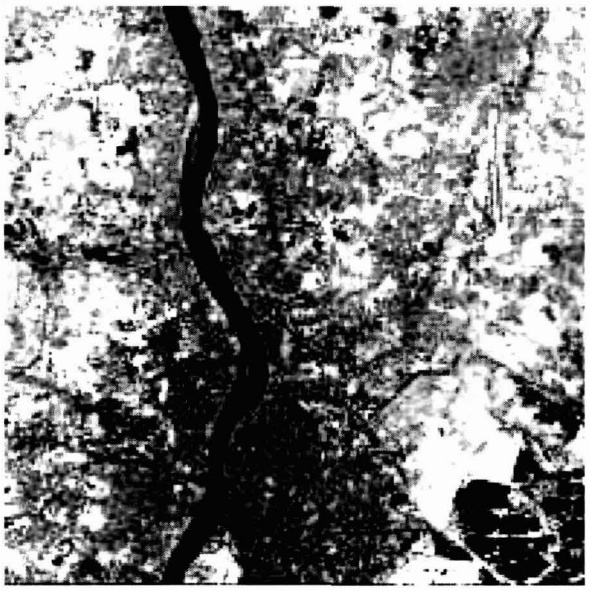


Figure 5.14: DisClus output of Figure 5.13

classes: Water Bodies (black color), Habitation and City area (deep gray color), Open space (light gray color) and Vegetation (white color). The river *Hoogly*, stadium, fisheries, city area as well as the airport runway is distinctly discernible in the output image. The predominance of city area on both sides of the river, particularly at the bottom part of the image is also correctly classified which corresponds to the central part of Kolkata city. Figure 5.15 shows the Kolkata image partitioned using FCM algorithm. It can be seen from the result that the river Hoogly and the city area has not been properly classified. These two objects have been classified as belonging to the same class. Similarly, the whole Salt Lake city as a whole has been put into one class. However, some portions such as canals, the Dumdum airport runway, fisheries, etc. have been classified properly.

The experiments on the images presented next is aimed to handle two different types of terrains (plain and hilly) in order to see the variation of classification accuracy. Dataset 3 shows the plain built up area of Sonari in Sibsagar district of Assam (Figure 5.16).

Some characteristic regions in the image are the river *Brahmaputra* shown in black

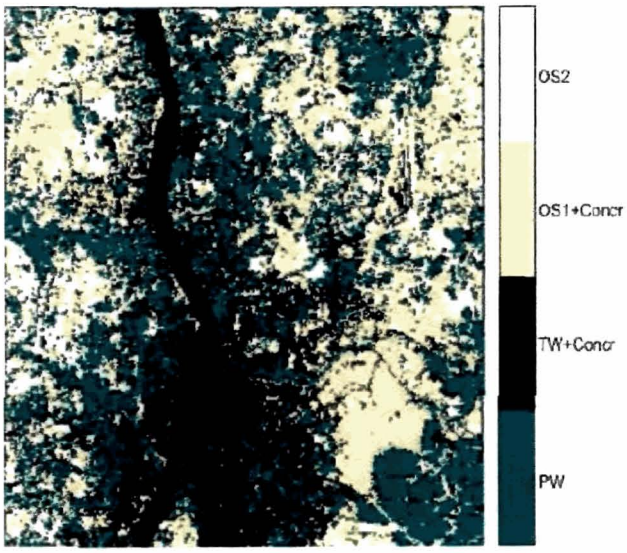


Figure 5.15: FCM output

Figure 5.16: Cartosat-1 of Sonari

Figure 5.17: DisClus output of Figure 5.16

color and spirally cutting across the middle of the image, roads, agricultural land, human settlements, etc. The DisClus clustering algorithm automatically detects 5 clusters (Figure 5.17 corresponding to river, road, agricultural land, water bodies and human settlements.

The fourth dataset used in this work shows a view of the Borapani area of the state of Meghalaya (Figure 5.18). The characteristic regions in this image are the Deep water (Deep Blue color), Wetlands (light blue color), Vegetation (Red and Pink colors) and Open spaces (White color).

DisClus clustered the image into five classes as shown in Figure 5.19. The resulting image classified the regions as: deep water (dark blue), wetland (sky blue), vegetation (pink), open spaces (white) and pond water (black). It can be seen that the water body at the left hand top corner of the image has been detected which corresponds well to the ground information available.

From the experimental results given above, we can conclude that the technique is highly capable of detecting clusters of all shapes.

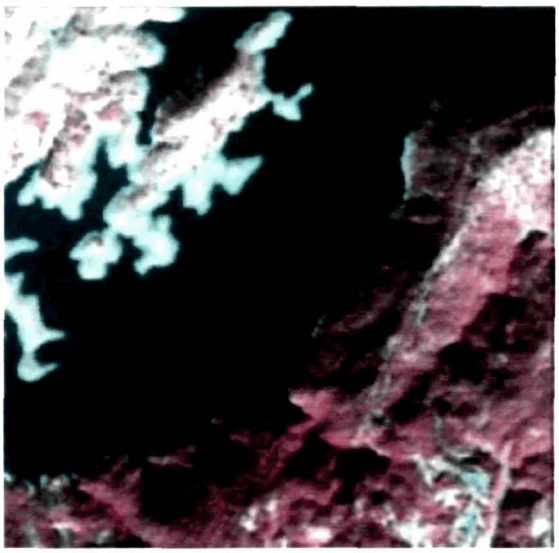


Figure 5.18: IRS image of Borapani

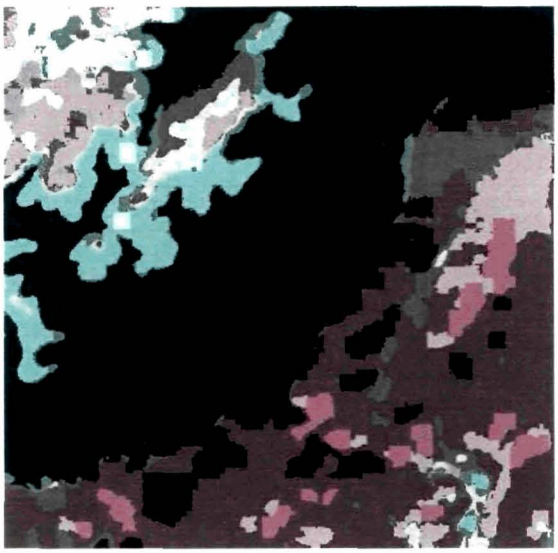


Figure 5.19: DisClus output of Figure 5.18

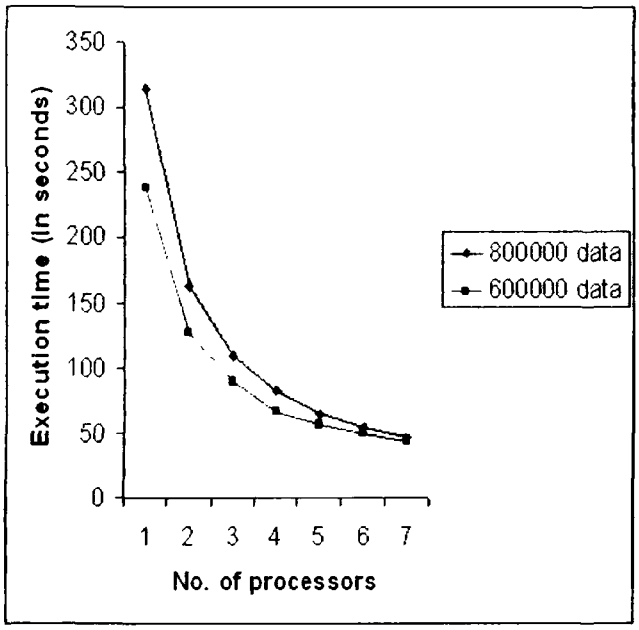


Figure 5.20: Execution time

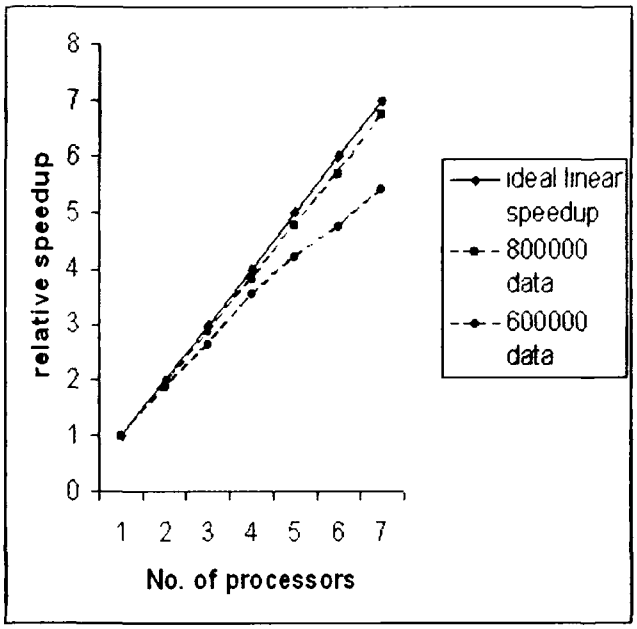


Figure 5.21: Relative Speedup curves

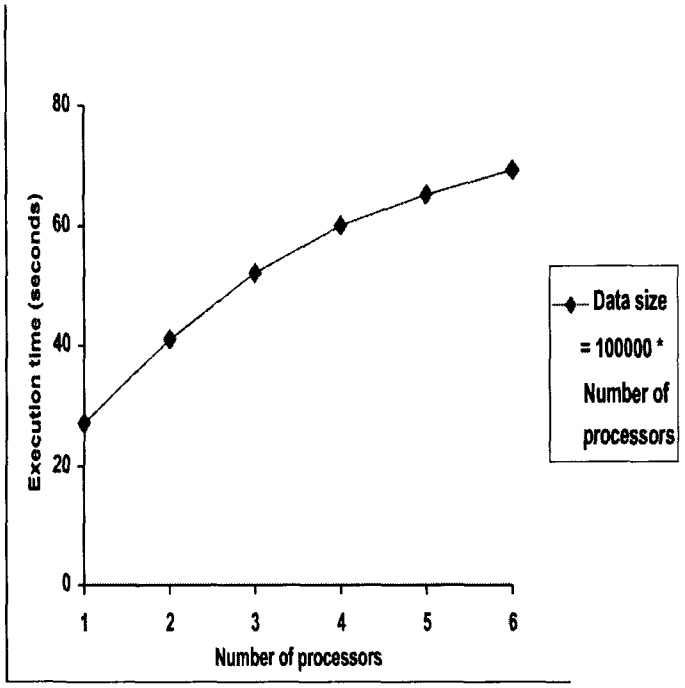


Figure 5.22: Scale-up curve

5.4.4 Performance and Scalability Analysis

In our implementation environment, there is no inter-processor communication except for a single processor communicating with each of the remaining processors. Each processor has the same specification i.e. PIV with 1 GHz speed and 128 MB RAM and the processors are connected through Ethernet LAN of speed 10/100 Mbps. To smooth out any variation, each experiment was carried out for five times and the average results were taken and each reported data point is to be interpreted as an average over five measurements. Our algorithm was implemented in JAVA in Linux environment in a HP xw8600 WS.

i. Parallel Execution Time: $T(k_p)$, the parallel execution time of a program is the time required to run the program on k_p nodes in parallel. When $k_p = T(1)$ denotes the sequential run time of a program on a single processor. Figure 5.20 reveals that the execution time decreases significantly with the increase in the number of processors.

ii. Speedup: Speedup is a measure of relative performance between a multiprocessor system and a single processor system, defined as, $S(k_p) = T(1)/T(k_p)$. On experimenting it has been found that the speedup factor increases with the increase in the number of processors. Figure 5.21 shows relative speedup curves for two datasets with points $N = 8{\times}10^5$ and $6{\times}10^5$. The number of dimensions and the number of clusters are fixed for both the datasets. The solid line represents "ideal" linear relative speedup. For each dataset, a dotted line connects observed relative speedups, which is a sub-linear type.

iii. Efficiency: The efficiency of a program on k_p processors, i.e. $E(k_p)$ is defined as the ratio of speedup achieved and the number of processors used to achieve it. $E(k_p) = S(k_p)/k_p = T(1)/k_p.T(k_p)$. In case of the proposed technique we observed that too many processors does not ensure the efficiency.

iv. Scale-up: The scale-up characteristic of the proposed technique has been found to be satisfactory with the increase in the number of processors as can be seen from Figure 5.22. Here the number of data points is scaled by the number of processors while dimensions and number of clusters are held constant.

While comparing to DBSCAN, OPTICS, EnDBSCAN, GDLC and Density-isoline, the proposed DisClus requires only two parameters i.e. the number of grid cells, i.e. gr_n and threshold α. However, based on our extreme experimental studies, it has been observed that the threshold α does not vary significantly with different datasets.

5.4.5 Comparison of Cluster Quality of DisClus with its Standalone Counterparts

The results of clustering the remote sensing images have been evaluated quantitatively using an index, β as in [PGS00]. Let n_i be the number of pixels in the i^{th} cluster ($i = 1, \cdots, c$), X_{ij} be the vector (of size 3×1) of the HSI values of the j^{th} pixel ($j = 1, \cdots, n_i$) for all the images in cluster i, and $\bar{X}_i$ the mean of n_i HSI values

of the i^{th} cluster. Then, β is defined as [PGS00]:

$$\beta = \frac{\sum_{i=1}^{c}\sum_{j=1}^{n_i}(X_{ij} - \bar{X})^T(X_{ij} - \bar{X})}{\sum_{i=1}^{c}\sum_{j=1}^{n_i}(X_{ij} - \bar{X}_i)^T(X_{ij} - \bar{X}_i)}$$

where n is the size of the image and $\bar{X}$ is the mean HSI value of the image. It may be noted that X_{ij}, $\bar{X}$, and $\bar{X}_i$ are all 3×1 vectors. The above measure is the ratio of the

Table 5.2: Comparison of β values for different clustering algorithms

Method	k-means [McQ67]	Astrahan's [Ast70]	Mitra's [MMP02]	SATCLUS	GDSDC	DisClus
β	5.30	7.02	9.88	17.82	12.63	15.31

total variation and within-cluster variation and is widely used for feature selection and cluster analysis [MMP02]. For a given image and c (number of clusters) value, the higher the homogeneity within the segmented regions, the higher the β value. The proposed DisClus has the highest β as can be seen in Table 5.2. DisClus was also compared with its other stand-alone and density based counterparts in terms of general parameters and the result is shown in Table 5.3.

5.5 Discussion

This chapter presents two clustering techniques: the first one (DGDCT) is for massive 2D spatial data and the second one is for satellite data. DGDCT is based on a grid-density based approach and can detect global as well as embedded clusters qualitatively. Experimental results of DGDCT in terms of scale-up and speedup are reported to establish the superiority of the technique in light of several synthetic datasets.

DisClus is also a grid-density based clustering technique for high-resolution multispectral satellite image. The technique was experimentally evaluated and found capable in detecting the clusters qualitatively. Experimental results establish the efficiency of the technique in light of several satellite images. In DisClus, there is also an option for choosing either the partition based algorithm (SATCLUS) or the fuzzy

Table 5.3: Comparison of DisClus with its counterparts

Algorithms	*No. of parameters*	Structure	*Complexity* (Approximate)
k-means	1 (N)	Spherical	$O(N)$
FCM	1 (N)	Non-Convex	$O(N)$
DBSCAN	2 $(MinPts, \varepsilon)$	Arbitrary	$O(NlogN)$ using $R*$ tree
OPTICS	3 $(MinPts, \varepsilon, \varepsilon\prime)$	Arbitrary	$O(NlogN)$ using $R*$ tree
SATCLUS	2 (gr_n, α)	Arbitrary	$O(k \times r)$
GDSDC	2 (gr_n, α)	Arbitrary	$O(k \times r)$
DisClus	2 (gr_n, α)	Arbitrary	$O(N)$

based one (GDSDC) for the clustering process depending on the image data. Results of both the algorithms have been reported in Chapter 4 to show their efficiencies. Since satellite images are huge in size, DisClus helps in handling such data efficiently and qualitatively. Moreover, DisClus uses the number of grid cells and threshold α as input parameters, however, it has been seen that α does not vary much with different datasets. The next chapter deals with the application of clustering over gene expression datasets.

Chapter 6

Clustering Gene Expression Data for Coherent Pattern Identification

This chapter presents two clustering methods capable of identifying coherent patterns over gene expression data. The first method, GenClus, is a technique for clustering gene expression datasets, designed based on a density based approach. It is capable of identifying clusters and sub-clusters of arbitrary shapes of any gene expression dataset even in presence of noise. Experimental results show the efficiency of GenClus in detecting quality clusters over gene expression data in terms of the z-score cluster validity measure. An incremental version of GenClus (InGenClus) is also presented that has been established to be effective in handling datasets that are updated incrementally.

The second method presents an effective tree-based clustering technique (*Gene ClusTree*) for finding clusters over gene expression data. GeneClusTree works by finding the maximal space clusters and then proceeds in finding the reduced space clusters. The clusters are represented as a tree with the reduced space clusters as the child of its respective maximal space cluster. The p-value analysis of GeneClusTree shows that it is capable in detecting biologically relevant clusters from gene expression data.

6.1 Introduction

Microarrays are a powerful technology that enables the monitoring of the expression levels of thousands of genes across different developmental stages, clinical conditions or time points. It helps in understanding gene functions, biological processes, gene networks, effects of medical treatments, etc.

The central dogma of bioinformatics describes the unidirectional flow of information from DNA via RNA (Ribonucleic acid) to protein in three steps: *Replication*, *Transcription* and *Translation* [KW02]. The first stage, *Replication* is the process which results in the duplication of the genetic information coded in DNA strands. The second stage, *Transcription*, is the transfer of information from the double stranded DNA into single-stranded mRNA. The third stage, *Translation*, refers to the conversion inside the cell where mRNA is translated to produce a protein. Together *Transcription* and *Translation* constitute *Gene Expression*. Gene expression experiments provide a method to quantitatively measure the transcription phase of protein synthesis. The objective of gene expression experiments is the quantitative measurement of mRNA expression particularly under the influence of drug or disease perturbations.

The two major types of microarray experiments are: cDNA microarray and oligonucleotide arrays. Though both the types of experiments follow different protocols, yet they have some common basic procedures [Ste06]. The analysis of gene expression is shown in Figure 6.1. The basic building blocks used in the analysis pipeline are described in brief below.

i *Chip manufacture*: A DNA microarray is a small chip consisting of a solid surface (made of chemically coated glass, nylon membrane or silicon), onto which DNA molecules (probes) have been chemically bonded in fixed grids. The purpose of a microarray is to detect the presence and abundance of labeled nucleic acids in a biological sample, which will hybridize to the DNA on the

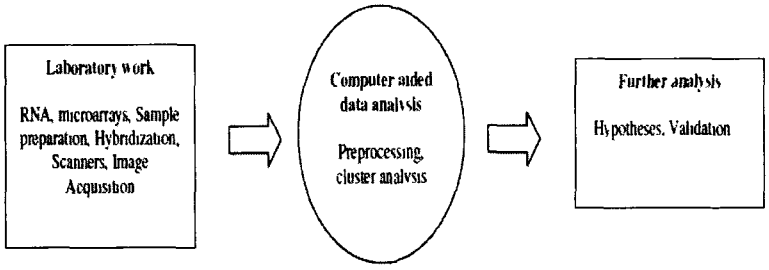


Figure 6.1: Gene expression analysis pipeline

array and can be detected by the label [Ste06]. The labeled nucleic acids are derived in the mRNA of a sample and hence microarray measures the gene expression. Since thousands of DNA molecules are bonded to a single array, it is possible to measure the expression of many thousands of genes in parallel.

ii *Sample preparation, labeling*: The first step is the extraction of RNA from the tissue of interest. Next, two mRNA samples are reverse-transcribed into cDNA and labeled using fluorescent dyes (Cy3 and Cy5).

iii *Hybridization and washing*: Hybridization is the step in which the DNA probes on the glass and the labeled DNA (or RNA) target form heteroduplexes via Watson-Crick base pairing [KW02]. After hybridization, the slides are washed (using a low-salt wash or with a high-temperature wash) to remove excess hybridization solution from the array. This ensures that only the labeled target on the array is the target that has specifically bound to the features on the array. This step also reduces cross-hybridization.

iv *Image Acquisition*: In this step, an image of the surface of the hybridized array (chip) is produced by scanning the chip to read the signal intensity that is emitted from the fluorescent dye of the heteroduplexes on the array where the target has bound to the probe. Raw data is obtained from this step.

Next, various normalization and standardization steps are performed to clean and filter the data and resolve any errors, noise and bias introduced by the microarray experiments. Finally, the real-valued gene expression data is obtained in the form of a matrix where the rows refer to the genes and the columns represent the conditions.

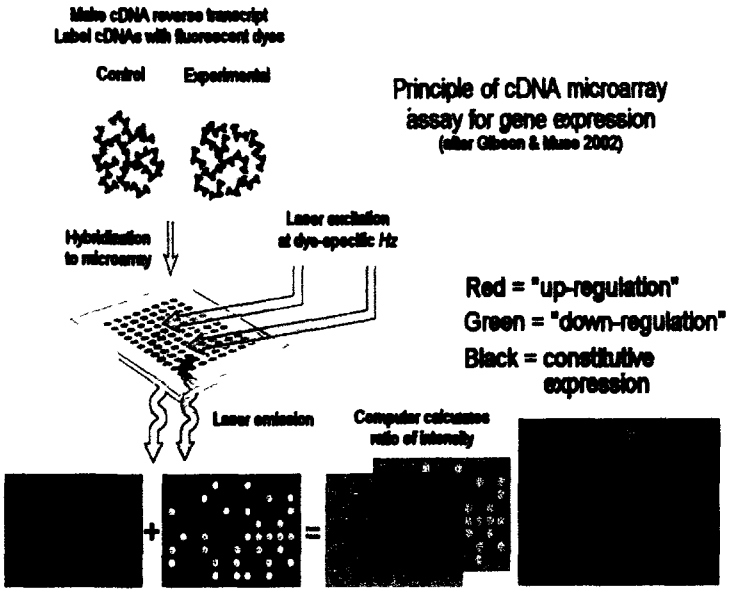


Figure 6.2: The image acquisition process

The next step is to use data mining techniques (such as clustering, association rule mining, etc) to extract the hidden information in this data. Finally, validation is performed to check if the result obtained is good from a biological point of view. The image acquisition process is shown in detail in Figure 6.2 and has been reproduced from *http://www.mun.ca/biology/scarr/cDNA_microarray_ Principle.jpg.*

The power of a microarray is that there may be many thousands of different DNA molecules bonded to an array, and so it is possible to measure the expression of many thousands of genes simultaneously.

6.2 Gene Expression Data

Gene expression is the effective production of the protein that a gene encodes. A microarray experiment assesses a large number of DNA sequences (genes) under multiple conditions (such as time-series, tissue samples (e.g., normal versus cancerous tissues), experimental conditions, etc.). A gene expression dataset from a microarray experiment may be considered as a $G \times T$ matrix D_G as shown in Equation 6.1,

where $D_G = \{g_{i,j}\}$, the rows of which represent expression patterns of a set of G genes $\{g_1, \cdots, g_G\}$, the columns represent expression profiles of a set of T samples, $S = \{s_1, \cdots, s_T\}$ and each cell $g_{i,j}$ is the expression level of gene $g_i (where\ 1 \leq i \leq G)$ on sample $s_j (where\ 1 \leq j \leq T)$.

$$D_G = \begin{bmatrix} g_{11}\ g_{12} \cdots\cdots g_{1T} \\ g_{21}\ g_{22} \cdots\cdots g_{2T} \\ \vdots \\ g_{G1}\ g_{G2} \cdots\cdots g_{GT} \end{bmatrix} \tag{6.1}$$

The main advantages of the analysis of gene expression data are: (i) meaningful patterns in the data are identified i.e. genes with unique properties are clustered and (ii) specific genes belonging to a pattern as well as the associations among groups of genes are identified. However, the large number of genes and the complexity of biological networks pose a serious challenge in interpreting the resulting data, which often consists of millions of measurements. This challenge can be addressed by the use of clustering techniques, which reveals natural structures and identifies the interesting patterns in the underlying data. A number of clustering methods have emerged for the analysis of gene expression data. Cluster analysis starts with this gene expression matrix and finds proximity between the different genes. Clustering algorithms group genes which are similar based on the proximity measure into the same cluster. Therefore, similar genes are grouped into the same cluster and dissimilar genes are grouped into different clusters. During the last couple of years, several significant coherent pattern identification techniques have been evolved under the categories of gene based, sample based and subspace clustering approaches. Next subsection is dedicated to reviewing some of those popular algorithms. According to [Ste06], most data mining algorithms developed for microarray gene expression data deal with the problem of clustering. Clustering groups genes with similar expression patterns into the same cluster. One of the characteristics of gene expression data is that it is meaningful to cluster both in terms of genes or samples. Co-expressed genes can be grouped into clusters based on their expression patterns ([BDSY99], [ESBB98]). Two major challenges for clustering gene expression data are: (i) to group genes with similar expression patterns (co-expressed genes), and (ii) to extract the useful patterns intelligently from noisy datasets.

6.3 Coherent Pattern Identification in Gene Expression Data

In this section, we report some of the popular gene based, sample based and subspace clustering methods, for identifying coherent patterns in gene expression data.

6.3.1 Gene based Clustering Approach

The goal of gene-based clustering is to group co-expressed genes together. Co-expressed genes indicate co-function, co-regulation and reveals the natural data structures [JTZ04]. In *gene-based* clustering, the genes are treated as the objects, while the samples are the features. Gene expression data consists of a high level of background noise. Therefore, clustering algorithms for gene expression data should be capable of extracting useful information from such noisy data and should depend as little as possible on prior knowledge. Also it is an added advantage if the clustering algorithm provides a graphical representation of the cluster structure other than just partitioning the data.

6.3.2 Sample based Clustering Approach

In *sample-based* clustering, the samples can be partitioned into homogeneous groups where the genes are regarded as features and the samples as objects. Samples are generally related to various time points, disease or drug effects within a gene expression matrix. According to [JTZ04], the goal of *sample-based* clustering is to find the "phenotype structures or substructures of the samples". Only a small subset of genes whose expression levels strongly correlate with the class distinction, rise and fall coherently and exhibits fluctuation of a similar shape under a subset of conditions, participate in any cellular process and are relevant. These genes are called the informative genes. The remaining genes are regarded as noise in the data and are irrelevant to the sample of interest. By focusing on a subset of genes and conditions of interest, the noise levels induced by other genes and conditions can be lowered which are characterized by co-clustering. Therefore, to identify informative genes and reduction of gene dimensionality for clustering samples to detect their substructure

particular methods should be applied. The *sample-based* clustering techniques are divided into two main categories: (i) clustering based on supervised informative gene selection and (ii) unsupervised clustering and informative gene selection. Since the percentage of the informative genes is very less, the major challenge of sample-based clustering is informative gene selection. In Supervised informative gene selection techniques, the sample's phenotype information is used to select informative genes. It is relatively easy to use and gets a high clustering accuracy rate since the majority of the samples are used as the training set to select informative genes. Unsupervised sample-based clustering as well as informative gene selection is quite difficult due to the fact that no prior knowledge is supposed to be known in advance. There are two strategies to address this problem. The first strategy reduces the number of genes before clustering samples and the second strategy utilizes the relationship between the genes and samples to perform gene selection and sample clustering simultaneously in an iterative paradigm [JTZ04]. One drawback of these approaches is that the gene filtering process is non-invertible. The deterministic filtering will cause data to be grouped based on local decisions. Also some knowledge about the number of clusters is needed which will be an input parameter of the clustering method. Another problem is determination of the number of iterations which usually is hard to estimate.

Both the gene-based and sample-based clustering approaches search for exclusive and exhaustive partitions of objects that share the same feature space. Apart from these, a third category, that is *subspace clustering*, captures clusters formed by a subset of genes across a subset of samples.

6.3.3 Subspace Clustering Approach

For subspace clustering algorithms, either genes or samples can be regarded as objects or features. To find subset of objects such that the objects emerge as a cluster in a subspace created by a subset of the features [JTZ04]. Genes and samples are treated symmetrically such that either genes or samples can be regarded as objects or features. A single gene may participate in multiple pathways that may or may not be co-active under all conditions. Subspace clustering [AGGR98] techniques confine

coherence exhibit by the blocks within gene expression matrices. A block is a sub-matrix defined by a subset of genes on a subset of samples. A subspace clustering algorithm, CLIC, has been proposed in [YHCY10]. CLIC first clusters the genes in individual dimensions and the ordinal labels of clusters in each dimension are then used for further full dimension-wide clustering. CLIC also finds the sub-clusters of the clusters detected in the first round of clustering which helps in finding more homogeneous groups in the data. Subspace clustering algorithms can be further subdivided into biclustering and triclustering algorithms as discussed next.

Biclustering: Biclustering [MO04] performs simultaneous clustering on the row and column dimensions of the data matrix. Simultaneous gene-condition (row-column) clustering is performed to identify sub-matrices, sub-groups of genes and subgroups of conditions. Clustering derives a global model while biclustering produces a local model. Unlike clustering algorithms, biclustering algorithms identify groups of genes that show similar activity patterns under a specific subset of the experimental conditions, each gene and condition in a bicluster are only a subset of the genes and conditions. In biclustering, if some points are similar in several dimensions they will be clustered together in that subspace. Biclustering has been proved of great value in finding the interesting patterns in the microarray expression data.

Triclustering: Triclustering [ZZ05] is mining coherent clusters in three-dimensional (3D) gene expression datasets. It mines arbitrary positioned and overlapping clusters and depending on different parameter values it mines diverse variety of clusters. It can detect clusters with constant or similar values along each dimension as well as scaling and shifting expression patterns. Tricluster relies on graph-based approach to mine all valid clusters and merge/delete some clusters having large overlaps and has been found to detect significant triclusters in the real microarray datasets. In [LT09], the authors designed a triclustering algorithm which utilizes a divide-and-conquer strategy, and an Automatic Boundary Searching (ABS) algorithm that is capable to detect statistically significant Regulated Expression Values [LT09] that correspond to a tricluster. A parallel version of tricluster algorithm is reported in [AFO+08].

Gene expression data has certain special characteristics and is a challenging research problem. In this chapter, we will mainly focus on gene-based clustering as we are interested in finding the co-expressed genes which will indicate co-function, co-regulation and reveal the natural structures in the data. Here, we will first present the challenges of gene-based clustering and then review a series of gene-based clustering algorithms.

6.3.4 Challenges of Gene-based Clustering

The purpose of clustering gene expression data is to reveal the natural structure inherent in the data. A good clustering algorithm should depend as little as possible on prior knowledge, for example, requiring the pre-determined number of clusters as an input parameter. Clustering algorithms for gene expression data should be capable of extracting useful information from noisy data. Gene expression data are often highly connected and may have intersecting and embedded patterns [JPZ03]. Therefore, algorithms for gene-based clustering should be able to handle this situation effectively. Finally, biologists are not only interested in the clusters of genes, but also in the relationships (*i.e.*, closeness) among the clusters and their sub-clusters, and the relationship among the genes within a cluster (e.g., which gene can be considered as the representative of the cluster and which genes are at the boundary area of the cluster). A clustering algorithm, which also provides some graphical representation of the cluster structure is much favored by the biologists.

6.4 Gene Based Clustering Algorithms: A Selected Review

We now present a review of some selected gene based clustering algorithms.

k-means [McQ67] is a typical partition-based clustering algorithm which divides the data into pre-defined number of clusters in order to optimize a predefined criterion. The major advantages of it are its simplicity and speed, which allows it to run on large datasets. However, it may not yield the same result with each run of the algo-

rithm. Often, it can be found incapable of handling outliers and is not suitable to detect clusters of arbitrary shapes. A Self Organizing Map (SOM) [Koh95] is more robust than k-means for clustering noisy data. It requires the number of clusters and the grid layout of the neuron map as user input. Specifying the number of clusters in advance is difficult in case of gene expression data. Moreover, partitioning approaches are restricted to data of lower dimensionality, with inherent well-separated clusters of high density. But, gene expression datasets may be high dimensional and often contain intersecting and embedded clusters. QT (quality threshold) clustering [HKY99] is an alternative method of partitioning data, invented for gene clustering. It requires more computing power than k-means, but does not require specifying the number of clusters apriori, and always returns the same result when run several times. The distance between a point and a group of points is computed using complete linkage, i.e., as the maximum distance from the point to any member of the group [ESBB98]. A hierarchical structure can also be built based on SOM such as Self-Organizing Tree Algorithm (SOTA) [DC97]. Recently, several new algorithms such as [HVD01] and [THHK02] have been proposed based on the SOM algorithm. These algorithms can automatically determine the number of clusters and dynamically adapt the map structure to the distribution of data. Herrero et al. [HVD01] extend the SOM by a binary tree structure. At first, the tree only contains a root node connecting two neurons. After a training process similar to that of the SOM algorithm, the dataset is segregated into two subsets. Then the neuron with less coherence is split into two new neurons. This process is repeated level by level, until all the neurons in the tree satisfy some coherence threshold. Other examples of SOM extensions are Fuzzy Adaptive Resonance Theory (Fuzzy ART) [THHK02] which provide some approaches to measure the coherence of a neuron (e.g., vigilance criterion). The output map is adjusted by splitting the existing neurons or adding new neurons into the map, until the coherence of each neuron in the map satisfies a user specified threshold.

The drawbacks of k-means are the lack of prior knowledge of the number of gene clusters in a gene expression data which results in the altering of results in successive runs since the initial clusters are selected randomly and the quality of the attained

clustering has to be assessed. The drawbacks of SOM is that it is not effective since the main interesting patterns may be merged into only one or two clusters and cannot be identified.

Unweighted Pair Group Method with Arithmetic Mean (UPGMA), presented in [ESBB98], adopts an agglomerative method to graphically represent the clustered dataset. However, it is not robust in the presence of noise. In [ABN+99], the genes are split through a divisive approach, called the Deterministic-Annealing Algorithm (DAA). The Divisive Correlation Clustering Algorithm (DCCA) [BD08] uses Pearson's Correlation as the similarity measure. All genes in a cluster have highest average correlation with genes in that cluster. Hierarchical clustering not only groups together genes with similar expression patterns but also provides a natural way to graphically represent the dataset allowing a thorough inspection. However, a small change in the dataset may greatly change the hierarchical dendrogram structure. The drawbacks of this method are its high computational complexity, lack of robustness, vagueness of termination criteria and failure with large number of genes as datasets grow in complexity.

A density based cluster can be defined as a region over the gene space, in which the local density is higher than its surrounding region. To identify such a region, we need to calculate local densities of genes in space. The density of genes is governed by two factors: (a) the typical distances among the genes, and (b) the number of neighbors of a gene, indicative of the dimension in which the points are embedded. Density based clustering algorithms identify dense areas in the object space. Clusters are hypothesized as high density areas separated by sparsely dense areas. A kernel density clustering method for gene expression profile analysis is reported in [SZCS03]. It assumes no parametric statistical model and does not rely on any specific probability distribution. Hyper-spherical uniform kernels of variable radius are used and density estimate of the data points are found. The method is robust and less sensitive to outliers. However, accurate density estimation and assignment of cluster membership require multiple data points in near neighborhoods and thus density estimation is less accurate when cluster size is small. In [JPZ03], the au-

thors propose the Density-based Hierarchical Clustering method (DHC) that uses a density-based approach to identify co-expressed gene groups from gene expression data. It considers clusters as high dimensional dense areas where the genes are attracted to each other. DHC uses two-level hierarchical structures (attraction tree and density tree) to organize the cluster structure of the dataset. The attraction tree reflects relationships among genes in the dense area. Each node in the attraction tree represents a gene and its parent is the attractor of it. The highest density gene becomes the root of the tree. The attraction tree becomes complicated for large datasets and hence the cluster structure is summarized in a density tree. Each node of the density tree represents a dense area. Initially the whole dataset is considered a single dense area represented by the root node of the density tree. This dense area is then split into several sub-dense areas based on some criteria where each sub-dense area is represented by a child node of the root node. The sub-dense areas are further split till each sub-dense area contains a single cluster. DHC is suitable for detecting highly connected clusters but is computationally expensive and is dependent on two global parameters. An alternative to this is to define the similarity of points in terms of their shared nearest neighbors. This idea was first introduced by Jarvis and Patrick [JP73]. In [CJM04], a k-nearest neighbor based density estimation technique has been exploited. The density based algorithm proposed by [CJM04] works in three phases: density estimation for each gene, rough clustering using core genes and cluster refinement using border genes. Density of a gene is calculated by the sum of similarities among its k nearest neighbors. Core genes are high density genes and the method proceeds by clustering core genes to form the rough clusters. Once the rough clusters are formed, the border genes are assigned to the most relevant cluster. In [SAP06], the authors present a density and shared nearest neighbor based clustering method. The similarity measure used is that of Pearson's correlation and the density of a gene is given by the sum of its similarities with its neighbors. The shared nearest neighbors of the dense genes are found and merged into the same cluster. The merging is done efficiently using a data structure called the P-tree [Per01]. In [DBK09a], a density based method (RDClust) is presented for clustering gene expression data using a two-objective function. The method uses regulation information as well as a suitable dissimilarity measure to cluster genes

into regions of higher density separated by sparser regions. Density based approach give clusters of good quality but suffers from input parameter dependency and high computational complexity with increase in dimensionality.

The Expectation Maximization (EM) algorithm [DLR77] is a model based algorithm and it discovers good values for its parameters iteratively. It can handle various shapes of data, but can be very expensive since a large number of iterations may be required. In [TH09], a signal shape similarity method is used to cluster genes using a Variational Bayes Expectation Maximization algorithm [BG03]. An important advantage of model-based approach is that it provides an estimated probability that a data object will belong to a particular cluster. Thus, a gene can have high correlation with two totally different clusters. Gene expression data are typically highly-connected; there may be instances in which a single gene has a high correlation with two different clusters. Thus, the probabilistic feature of model-based clustering is particularly suitable for gene expression data. However, model-based clustering relies on the assumption that the dataset fits a specific distribution which may not be true in many cases.

Among the graph based algorithms, the CLuster Identification via Connectivity Kernels (CLICK) method ([SS00]) is suitable for subspace and high dimensional data clustering. CLICK is robust to outliers and does not make assumptions about the number or structure of clusters. Although CLICK does not need the number of clusters apriori, the algorithm may generate a large number of clusters because of the use of a homogeneity parameter. Ben-Dor introduced the idea of *corrupted clique graphs* and used the concept of a clique graph and divisive clustering in his algorithm, Cluster Affinity Search Techniques (CAST) [BDSY99]. A Clique graph is an undirected graph formed by the union of disjoint complete sub-graphs where each clique represents a cluster. The model assumes that there is a *true biological partition of the genes into disjoint clusters based on the functionality of genes* [BDSY99]. The genes (objects) form sub-graphs or cliques where intra-clique genes are completely similar and inter-cluster genes are completely dissimilar. CAST takes as input the pairwise similarities between genes and an affinity threshold, t. The algorithm searches

through the clusters one at a time adding to or removing genes from a cluster w.r.t. a connectivity condition. CAST does not require a user-defined number of clusters and is capable of handling outliers efficiently. But, it faces difficulty in determining a good threshold value. In CAST, the size and number of clusters produced is directly affected by the fixed user-defined parameter, affinity threshold, t. Hence, apriori domain knowledge of the dataset is required. To overcome this problem, E-CAST ([BPC02]) calculates the threshold value dynamically based on similarity values of the objects that are yet to be clustered. The threshold is computed at the creation of each cluster. The graph theoretic approach can be considered to be relevant to gene expression data mining as they are capable of discovering intersected clusters. However, it sometimes generates non-realistic cluster patterns.

Fuzzy c-means [Bez81b] and Genetic Algorithms (GA) (such as [BMM07b], [Gol89] and [MMB09]) have been used effectively in clustering gene expression data. The Fuzzy c-means (FCM) algorithm [Bez81b] when applied to gene expression data links each gene to all clusters via a real-valued vector of indexes. The values u_{ki} of the components of this vector lie between 0 and 1. For a given gene, an index close to 1 indicates a strong association to the cluster. Inversely, indexes close to 0 indicate the absence of a strong association to the corresponding cluster. The vector of indexes thus defines the membership of a gene with respect to the various clusters. Membership vector values u_{ki} and cluster centroids c_k can be obtained after minimization of the total inertia criterion [Bez81b]. The FCM algorithm requires the specification of two parameters, k i.e., the number of clusters in the dataset and m i.e., the fuzziness parameter. In [DK03], an empirical method, based on the distribution of distances between genes in a given dataset, is proposed to determine an adequate value for m. In [TBK09], the authors propose a novel semi-supervised clustering method called GO Fuzzy c-means, which is based on the fuzzy c-means clustering algorithm and utilizes the Gene Ontology annotations as prior knowledge to guide the process of grouping functionally related genes. Genetic algorithms [Gol89] have been extensively used to develop efficient clustering techniques. These techniques use a single cluster validity measure as the fitness function to reflect the goodness of an encoded clustering. However, a single cluster validity measure is seldom equally applicable

for different kinds of datasets. GA based methods have been applied over gene expression data and good results were obtained [MMB09], [BMM07b]. In [MMB09], a fuzzy majority voting approach is proposed that first identifies the genes which are assigned to some particular cluster with high membership degree by most of the Pareto-optimal clustering solutions. Using this set of genes as the training set, the remaining genes are classified by Support Vector Machine (SVM) classifier. A two-stage clustering algorithm employing, (i) variable string length genetic scheme [MB03b] and (ii) multiobjective genetic clustering has been proposed in [BMM07b]. The method is based on the concept of points having significant membership to multiple classes. For the clustering an iterated version of FCM is used. The GA based algorithms have been found to detect biologically relevant clusters but are dependent on proper tuning of the input parameters.

The current information explosion, fuelled by the availability of the World Wide Web and the huge amount of microarray experiments being conducted, have led to ever-increasing volume of data. Therefore, there is a need to introduce incremental clustering so that updates can be clustered in an incremental manner. Though a lot of research has been performed on incremental clustering in other application domains, incremental clustering of gene expression data has not been explored much.

Due to the huge number of microarray experiments being conducted regularly, whenever new gene expression data becomes available it is highly desirable to perform updates (i.e., incorporate the new results to existing clusters) with these newly arrived genes incrementally. In [EKS+98], the authors present an incremental clustering approach based on the DBSCAN [EKSX96] algorithm. Rough set theory has been employed in the incremental approach for clustering interval datasets in [ANS06]. It groups the given dataset into a set of overlapping clusters by employing a rough variant of the Leader algorithm [ANS06]. The algorithm generates cluster abstractions in a single scan and is robust to outliers. In [CCFM97], the authors present an incremental clustering model for information retrieval applications. [CHNW96] and [FAAM97] also report efficient methods for modifying a set of association rules.

In [LLF+04b], an incremental genetic k-means algorithm (IGKA) has been presented. IGKA calculates the objective value called Total Within-Cluster Variation (TWCV) and cluster centroids incrementally whenever the mutation probability is small. IGKA converges to the global optimum. In the Genetic k-means Algorithm (GKA) proposed in [KM99], a genetic algorithm is hybridized with the k-means algorithm and therefore GKA converges to the global optimum faster than other genetic algorithms. In [LLF+04a], the authors present a faster version of GKA (FGKA) that efficiently evaluates the TWCV, avoids illegal string termination overhead and simplifies the mutation operator. IGKA inherits all the advantages of FGKA and outperforms FGKA when the mutation probability is small. The cost of calculating the centroids in FGKA is more expensive when the mutation probability is smaller than when it is calculated incrementally in IGKA. The Hybrid Genetic k-means Algorithm (HGKA) in [LLF+04b] combines the advantages of both IGKA and FGKA and obtains an even better performance. However, it is very difficult to obtain the threshold value which is dataset dependent. In [RRAR06], an incremental gene selection algorithm (Best Incremental Ranked Subset (BIRS)) that reduces search space complexity using a wrapper-based method is presented. This method works on the ranking directly. In BIRS [RRAR06], genes are first ranked w.r.t. an evaluation measure. Then, the set of genes is updated by crossing the ranking from the beginning to the last ranked gene. Classification accuracy with the first gene in the list is obtained and it is marked as selected. The classification rate is again obtained and the second gene is selected depending on whether the classification accuracy is significantly better. The process is repeated till the last gene on the ranked list is processed. The algorithm returns the best subset formed and it does not contain irrelevant or redundant genes.

6.4.1 Discussion and Motivation

From our selected survey we conclude that clustering algorithms are useful in identifying groups of co-expressed genes and in discovering coherent expression patterns. Also it is observed that various clustering algorithms require different types of input parameters and clustering results are highly dependent on the values of the parameters. Majority of the clustering techniques are dependent on a choice of proximity

measure. Also, due to the inherent high dimensionality and presence of noise in gene expression data, it is a challenging task to find the clusters inherent in the subspaces of the dataset. Therefore, development of clustering techniques which are free from the restrictions offered by proximity measures, are independent of input parameters and are able to detect clusters embedded in the subspaces of gene expression data is of utmost importance. This chapter presents two clustering techniques (GenClus and GeneClusTree) for gene expression data which handles some of the challenges offered by gene expression data.

InGenClus is designed based on density based approach. It retains the regulation information which is also the main advantage of the clustering. It uses no proximity measure and is therefore free from the restrictions offered by them. An incremental version of GenClus (InGenClus) is also presented that can handle incremental data.

The hierarchical approach of clustering genes helps in visualizing the clusters at different levels of hierarchy. Also, it is important to establish that the clusters obtained are biologically relevant. We introduce an effective tree-based clustering technique (GeneClusTree), which is capable of identifying clusters of arbitrary shapes of any gene expression dataset, even in presence of noise. GeneClusTree attempts to find all the possible clusters over subspaces in minimum possible scans of the dataset.

6.5 GenClus

In this section, we introduce an effective gene-based clustering approach (GenClus), which is capable of identifying clusters and sub-clusters of arbitrary shapes of any gene expression dataset, even in presence of noise. GenClus attempts to find sub-clusters which may be relevant for biologists. The detection of sub-clusters provides the opportunity to uncover more homogeneous groups in the clusters. An advantage of GenClus is that it does not use any proximity measure during clustering the genes and is therefore free from the restrictions offered by various proximity measures. GenClus gives a hierarchical view of the clusters and sub-clusters formed. With the increasing development of internet technology and with the constant increase in the

microarray experimentation conducted, it has led to the ever-increasing volume of data. There is, therefore, a need to introduce incremental clustering so that updates can be clustered in an incremental manner. To handle such increase in volume of microarray data, incremental clustering technique often has been found suitable. This section also introduces an incremental version of GenClus *i.e.*, InGenClus which has been established to perform well in terms of several gene datasets.

Both GenClus and InGenClus can be found to be significant in view of the following points:

- provides a hierarchical cluster solution;
- free from the use of proximity measures;
- faster processing due to simplified matching mechanism;
- capable of handling noisy datasets;
- does not require the number of clusters apriori;

GenClus improves the quality of the clusters by identifying sub-clusters within large clusters. It can also handle the situation when the database is updated incrementally using less computation time.

6.5.1 Basics of GenClus

GenClus is a gene based clustering technique which adopts the notion of density based approach as can be found in [DBK10], [EKSX96]. It exploits a discretization technique which retains the up- or down- regulation information. Discretization is the process of putting values of a continuous set of data into buckets so that there are a limited number of possible values. The discretization of the dataset helps in keeping track of the regulation information of the data which is used later on in the clustering phase. GenClus normalizes the gene expression data and works over a discrete domain (of regulation information). Clustering is then run on the discretized data.

The gene expression data is normalized to have mean 0 and standard deviation 1. Expression data having a low variance across conditions as well as data having more than 3-fold variation are filtered. Discretization is then performed on this normalized expression data. Discretization uses the regulation information, i.e. up- or down-regulation in each of the conditions for a particular gene. Here, let G^* be the set of all genes and T^* be the set of all conditions. The discretization is done as follows:

i. The discretized value of gene g_i at condition, t_1 (i.e., the first condition)

$$\xi_{g_i,t_1} = \begin{cases} 1 & \text{if } \varepsilon_{g_i,t_1} > 0 \\ 0 & \text{if } \varepsilon_{g_i,t_1} = 0 \\ -1 & \text{if } \varepsilon_{g_i,t_1} < 0 \end{cases}$$

ii. The discretized values of gene g_i at conditions t_j $(j = 1,..(T-1))$ i.e., at the rest of the conditions $(T - \{t_1\})$

$$\xi_{g_i,t_{j+1}} = \begin{cases} 1 & \text{if } \varepsilon_{g_i,t_j} < \varepsilon_{g_i,t_{j+1}} \\ 0 & \text{if } \varepsilon_{g_i,t_j} = \varepsilon_{g_i,t_{j+1}} \\ -1 & \text{if } \varepsilon_{g_i,t_j} > \varepsilon_{g_i,t_{j+1}} \end{cases}$$

where ξ_{g_i,t_j} is the discretized value of gene g_i at condition t_j $(j = 1,..(T-1))$. The expression value of gene g_i at condition t_j is given by ε_{g_i,t_j}. We see in the above computation that the first condition, t_1, is treated as a special case and it's discretized value is directly based on ε_{g_i,t_1} i.e., the expression value at condition t_1. For the rest of the conditions the discretized value is calculated by comparing its expression value with that of the previous value. This helps in finding whether the gene is up- (1) or -down (-1) regulated at that particular condition. Each gene will now have a regulation pattern ($\wp$) of 0, 1, and -1 across the conditions or time points. This pattern is represented as a string.

Each gene is divided into various *range_ids* depending on their expression values as follows. The range_value for each expression level is given by uniformly dividing the difference between the maximum and minimum values in the normalized data.

$$range\_value = \frac{Max_{EV} - Min_{EV}}{interval} \tag{6.2}$$

Gene 1	Regulation	-1	1	1	1	1	-1	-1
	Range	0	0	0	1	1	0	0
Gene 2	Regulation	-1	1	1	1	1	-1	-1
	Range	0	0	0	1	1	0	0
Gene 3	Regulation	-1	1	1	1	1	1	1
	Range	0	0	0	0	1	2	3
Gene 4	Regulation	-1	1	1	1	1	1	1
	Range	0	0	0	0	1	2	3
Gene 5	Regulation	-1	1	1	1	1	1	1
	Range	0	0	0	0	1	4	4
Gene 6	Regulation	-1	1	-1	-1	1	-1	-1
	Range	0	0	0	0	0	1	1
Gene 7	Regulation	1	1	1	1	1	-1	-1
	Range	0	0	0	0	1	0	0
Gene 8	Regulation	1	1	1	1	1	-1	-1
	Range	0	0	0	1	1	-1	-1
Gene 9	Regulation	1	-1	1	-1	1	1	1
	Range	0	-1	0	-1	0	0	0

Figure 6.3: Example discretized dataset

where Max_{EV} is the maximum expression value and Min_{EV} is the minimum expression value. For example, suppose $interval = 7$. Therefore, we will have 7 *range_ids* (3, 2, 1, 0, -1, -2, -3), where the expression values of a gene falling in the corresponding range will get its *range_id*. Now, each gene will have a pattern of *range_ids* across the conditions or time points which is represented as a string. Figure 6.3 illustrates an example of a discretized matrix showing the regulation pattern and *range_ids*, where the number of intervals is set to 7, namely (3, 2, 1, 0, -1, -2, -3). The regulation information and range values are used together to cluster the gene expression dataset using a density based approach. By using these two values in combination as will be seen next, we do not need the use of any proximity measure. A string matching approach is used for matching the regulation pattern and range pattern of two genes. Next, we give some definitions which provide the foundation of GenClus.

Definition 22. Neighborhood level of a gene: A gene g_j is said to be a neighbor of gene g_i i.e., $g_j \in N_{level}(g_i)$ if (i) g_i matches with g_j over each of the v conditions, where v is greater than a user defined threshold, α; (ii) $range\_id(g_i, t_k) \pm level = range\_id(g_j, t_k)$, t_k refers to the conditions where $k = 1, 2, \cdots, T$ and *level* is a dynamically calculated parameter. (Initially, $level = 0$)

Definition 23. Core gene: A gene g_i is said to be a core gene if $\mid N_{level}(g_i) \mid \geq \sigma$ (user-defined threshold).

In our experiments we have obtained good results for $\sigma = 2$. Initially, level $= 0$ and the neighborhood of gene g_i is searched for genes satisfying the core gene condition of *Definition* 23. If no neighbor gene is found, then level is increased in both positive and negative range by one *i.e.*, we search for neighbor genes in adjacent *range_ids* of (g_i, t_k) and the neighborhood search continues.

Definition 24. Direct-Reachability: A gene g_j is said to be directly reachable from another gene g_i if g_i is a core gene and $g_j \in N_{level}(g_i)$.

Definition 25. Reachability: A gene g_j is said to be reachable from another gene g_i if there is a chain of genes $g_1, g_2, \cdots, g_p$ between g_i and g_j such that g_{i+1} is directly reachable from g_i.

Finding sub-clusters within bigger clusters gives the finer clustering of a dataset. Sub-cluster information may be useful for the biologists by means of visual display and in the interpretation.

Definition 26. Sub-Cluster: Let D_G be a database of genes. A sub-cluster S_i is a non-empty subset of D_G satisfying the following conditions:

1. $\forall g_i, g_j$: if $g_i \in S_i$ and g_j is reachable from g_i, then $g_j \in S_i$.

2. g_i matches with g_j over each of the v conditions.

3. $range\_id(g_i, t_k) \pm level = range\_id(g_j, t_k)$, t_k refers to the conditions where $k = 1, 2, \cdots, T$.

Definition 27. Cluster: Genes $g_i, g_j \in C_i$ (i^{th} cluster), if g_i matches with g_j over each of the v conditions *i.e.*, all genes having same regulation pattern over v conditions are grouped into the same cluster.

Sub-cluster S_j where, $j = 1, 2, \cdots$ will belong to cluster C_i, if it has the same regulation pattern w.r.t. C_i.

Definition 28. Noise Genes: Let $C_1, C_2, \cdots C_n$ be the set of clusters of D_G, then noise is the set of genes in D_G not belonging to any cluster C_i, *i.e.*,

$$noise = \{g_x \in D_G \mid \forall i : g_x \notin C_i\}$$

The clustering process starts with an arbitrary gene g_i and searches the neighborhood of it to check if it is core. If g_i is not core then the process restarts with another unclassified gene. If g_i is a core gene, then clustering proceeds with finding all reachable genes from g_i. All reachable genes are assigned the same sub_cluster_id as g_i. From the neighbors of g_i, if any gene satisfies the core gene condition, sub cluster expansion proceeds with that gene. The process continues till no more genes can be assigned to the sub cluster. The process then restarts with another unclassified gene and starts forming the next sub cluster. The clustering process continues till no more genes can be assigned sub_cluster_id. Once all sub clusters have been assigned, the process groups all sub-clusters as well as genes having no sub_cluster_id but having the same regulation pattern into the same cluster and assign them the same cluster_id. All unclassified genes are now termed as noise genes.

The clusters and sub-clusters for the example dataset of Figure 6.3 is illustrated in Figure 6.4. It can be observed that sub-clusters give the highly coherent patterns in the dataset. The algorithms for cluster formation and cluster expansion are given in Figure 6.5 and Figure 6.6. The experimental results of GenClus are reported in Section 6.5.3.

Microarrays generate tens of thousands of data in one experiment. Data volume is constantly increasing due to the huge amount of microarray experiments performed. While clustering this type of data, it is of utmost importance that the updations of the database are handled incrementally. Some of the incremental clustering algorithms have been reported in section 6.4. Though a lot of work has focused on incremental clustering over spatial datasets, not much research has been done on handling incremental gene expression data.

We therefore introduce an incremental clustering technique (InGenClus) for gene expression data, which is based on GenClus. Once clustering of the dataset is ob-

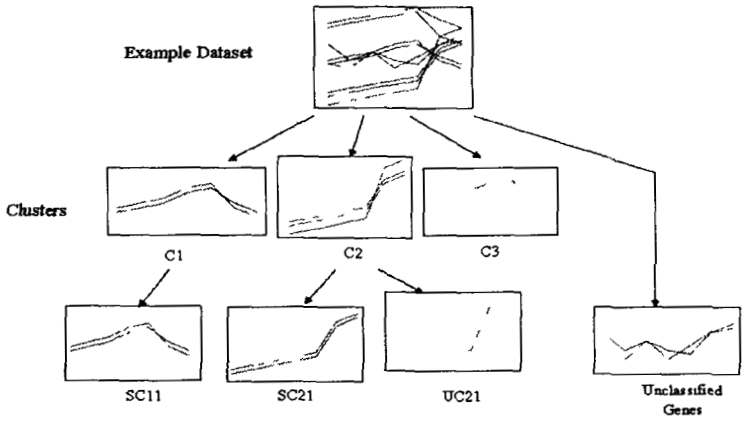


Figure 6.4: Clustering of the example dataset given in Figure 6.3. Here, C_is ($i = 1, 2, \cdots$) are clusters; SC_{ij} refer to the j^{th} sub-cluster of cluster i and UC_{ik} is the k^{th} gene in cluster i not belonging to any sub-clusters.

Cluster_creation()

//Pre-condition: All genes are unclassified
// $cluster\_id = 0$
  FOR i from 1 to D_G do
    IF g_i.classified = unclassified THEN
      Cluster_expand(g_i, $cluster\_id$)
      $cluster\_id$++;
    END IF
  END FOR

Figure 6.5: Algorithm for cluster formation of GenClus

Cluster_expand(g_i, *cluster_id*)

```
IF get_core(g_i) = 0 THEN
  g_i.cluster_id = cluster_id;
  RETURN;
ELSE
  g_i.classified = classified;
  FOR j from 1 to D_G do
    IF g_j.classified = unclassified
      Expand_cluster(g_j, cluster_id)
    END IF
  END FOR
END IF
```

Figure 6.6: Algorithm for cluster expansion of GenClus

tained, each of the clusters are represented by cluster profiles. The cluster profiles store the regulation of that particular cluster. The sub-clusters are represented by the sub-cluster profiles which stores the regulation and range information of that particular sub-cluster. This information is further used by InGenClus when clustering the updated database incrementally.

6.5.2 Incremental Clustering

In this section, we present InGenClus which is based on GenClus and is capable of handling incremental data. Due to the density based nature of GenClus, the insertion of a gene affects the current clustering only in the neighborhood of the gene. We examine the parts of an existing clustering affected by an update and show how InGenClus can handle incremental updates of a clustering after insertions.

The changes of the clustering of the gene database D_G are restricted to the neighborhood of an inserted gene. The previously core genes [DBK10] retain their core property but, non-core genes (border genes or noise genes) may become cores. Thus

new density connections may surface. The insertion of a gene g_i may result in a change of cluster membership of genes in the neighborhood of g_i and all genes reachable from one of these genes in $D'_G = D_G \cup \{g_i\}$, where D'_G is the updated dataset. While inserting g_i the following cases may occur:

1. *Fusion*: A gene g_i may be fused to a cluster C_i if regulation pattern of g_i matches with cluster profile of C_i, then g_i is fused into cluster C_i. Gene g_i may be fused to a cluster S_i if g_i is reachable from S_i.

2. *Cluster Creation*: Gene g_i may have same regulation pattern w.r.t. some other noise or unclassified gene(s) and may lead to the formation of a new cluster.

3. *Sub − cluster Creation*: Gene g_i may become core w.r.t. (i) some other noise or unclassified gene(s) and may lead to the formation of a new sub-cluster, (ii) some gene(s) in a cluster which are not members of any sub-cluster and this also lead to the creation of a new sub-cluster.

4. *Noise*: If g_i does not match with any of the cluster profiles then g_i is a noise gene and no density-connections are changed.

InGenClus starts with a newly inserted gene g_i and finds if its regulation and range information matches with any of the cluster or sub-cluster profiles then there can be the following cases:

i. g_i matches with cluster profile of C_i, then GenClus will assign *cluster_id* of C_i to g_i. After insertion of g_i, one of the genes $g_k \in C_i$ and $g_k \in S_i$ (S_i is a sub-cluster in C_i) may become core and hence can become a potential candidate for sub-cluster expansion (*case* 1).

ii. g_i matches with none of the cluster profiles, but it matches with some other unclassified genes. Then it creates a new cluster (case 2) and finds if it can form sub-clusters (case 3). Finally, it forms the cluster and/or sub-cluster profiles accordingly.

iii. g_i matches with cluster profile of C_i, then InGenClus will assign *cluster_id* of C_i to g_i. After insertion of g_i, any gene $g_k \in C_i$ and $g_k \notin$ any sub-clusters in C_i

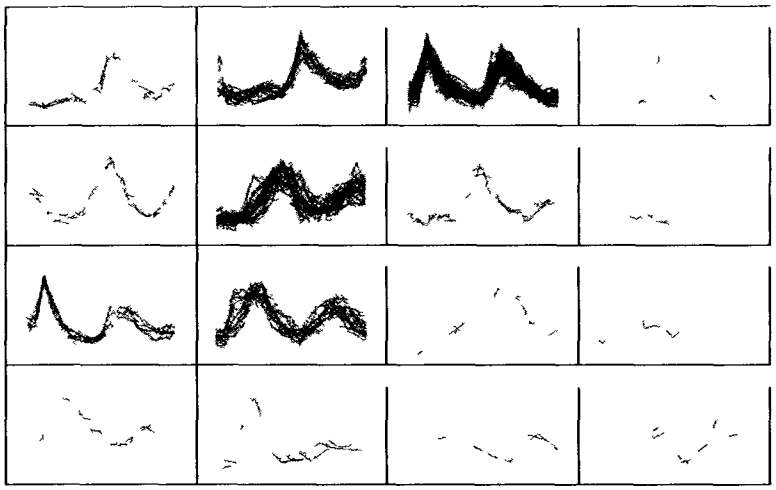


Figure 6.7: Some clusters are illustrated from the Dataset 1

may become core and hence may become a potential candidate for sub-cluster creation (*case* 3).

iv. g_i matches with none of the cluster profiles nor does it match with any other gene, then *case* 4 occurs.

In case of fusion, the affected cluster profiles are updated based on our own string matching technique. To achieve better space-time complexity, the cluster profiles are organized using an effective data structure. It has been found that the InGenClus yields the same result as when compared with GenClus, yet in a lesser time.

6.5.3 Performance Evaluation

GenClus was implemented in Java in Windows environment and evaluated with several real-life datasets as discussed next.

1. Datasets Used

- *Dataset 1*: In [CCW+98], Cho *et al.* used the temperature sensitive mutant strain CDC28-13 to produce a synchronized cell culture of the *Saccharomyces cerevisiae* from which 17 samples were taken at 10 minute intervals and hybridized to Affymetrix chips. The final data is publicly available

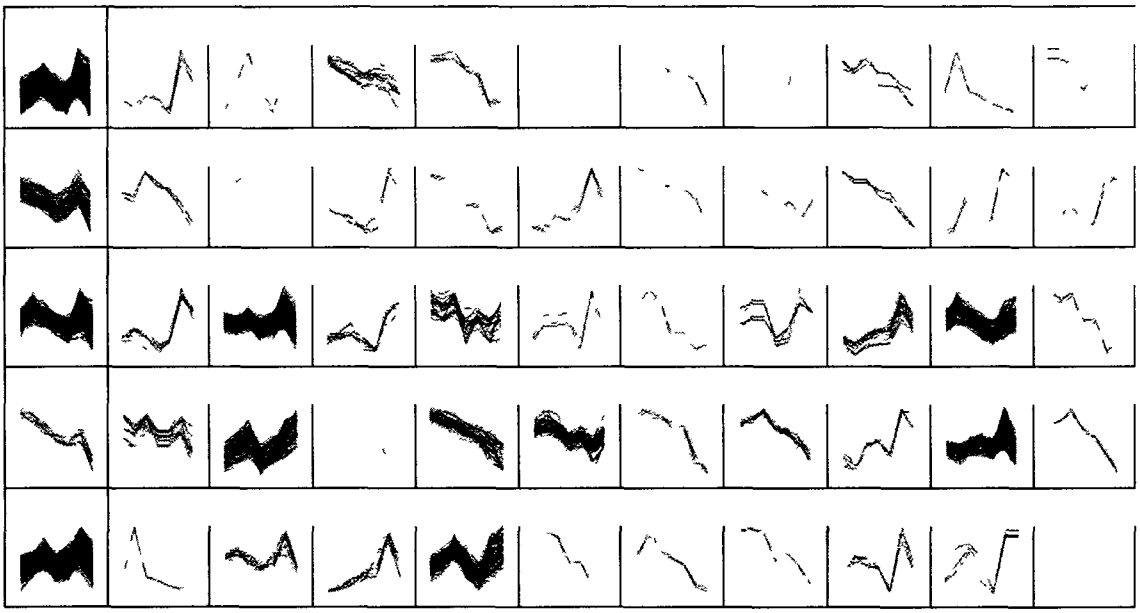


Figure 6.8: Some clusters are illustrated from the full Dataset 2

at *http://yscdp.stanford.edu/yeast_cell_cycle/full _data.html.* Cho's dataset is widely available and has functional classification that allows validation of clustering results. This dataset contains 6218 genes at 17 time points. Out of the full Dataset 1, a subset of 384 genes have been obtained from *http : //faculty.washington.edu/kayee/cluster.*

- *Dataset 2*: In [DI97], the authors use DNA microarrays to study the temporal gene expression of 6089 genes in *Saccharomyces cerevisiae* during the metabolic shift from fermentation to respiration. Expression levels were measured at seven time points during the diauxic shift (the two growth phases of a microorganism in batch culture as it metabolizes a mixture of two sugars). The full dataset can be downloaded from the Gene Expression Omnibus website, *http://www.ncbi.nlm.nih.gov/geo/query.*

- *Dataset 3*: The dataset describes the response of human fibroblasts to serum on cDNA microarrays in order to study growth control and cell cycle progression. These data were obtained from the study of [IER+99]. Primary cultured fibroblasts from human neonatal foreskin are induced to enter a quiescent state by serum deprivation for 48 hours and then stimulate by addition of medium

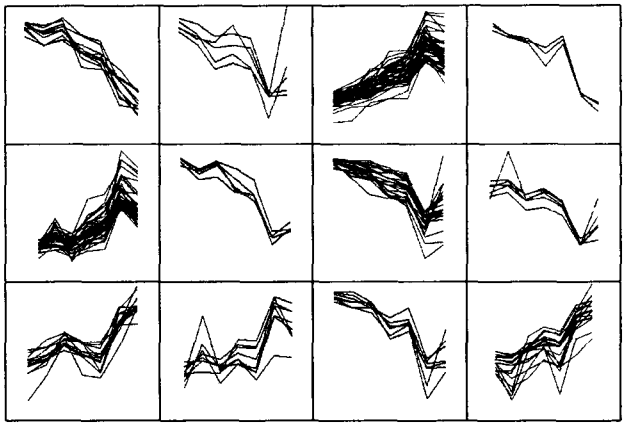


Figure 6.9: Result of GenClus on the reduced form of Dataset 2

containing 10% FBS. DNA microarray hybridization is used to measure the temporal changes in mRNA levels of 8613 human genes at 13 time points, ranging from 15 min to 24 hours after serum stimulation. In this thesis, we choose a subset of 517 genes whose expression changed substantially in response to serum. The detailed information about the dataset can be found at the Web site: http://genome-www.stanford.edu/serum/.

A brief overview of the datasets is given in Table 6.1. All the datasets are normalized to have mean zero and standard deviation one. Of the various datasets, the clusters obtained from the reduced Dataset 1 are shown in Figure 6.7. Some of the clusters formed from the full and reduced form of Dataset 2 are shown in Figure 6.8 and Figure 6.9. The datasets have been reduced by filtering out low variance genes and genes having more than 3-fold standard deviation. Some of the clusters obtained by GenClus from Dataset 3 are shown in Figure 6.10. The hierarchy of four of the clusters and sub-clusters of Dataset 2 is shown in Figure 6.11 where the full dataset is at the root, the clusters are shown with the single line frames, sub-clusters are shown with double line frames and the genes which are part of a higher level cluster but not part of any sub-clusters are shown with dotted line frames. In the figure, the full dataset is at the root, the clusters are shown with the single line frames, sub-clusters are shown with double line frames and the genes which are part of a higher level cluster but not part of any sub-clusters are shown with dotted line frames. The data from Dataset 2 was inserted incrementally and InGenClus was executed.

Table 6.1: Datasets used for evaluating the clustering algorithms introduced in this thesis

Serial No.	Dataset	No. of genes	No of conditions	Source
1	Yeast CDC28-13 [CCW+98]	6218	17	http://yscdp.stanford.edu/yeast_cell_cycle/full_data.html
2	Yeast Diauxic Shift [DI97]	6089	7	http://www.ncbi.nlm.nih.gov/geo/query
3	Subset of Human Fibroblasts Serum [IER+99]	517	13	http://www.sciencemag.org/feature/data/984559.hsl

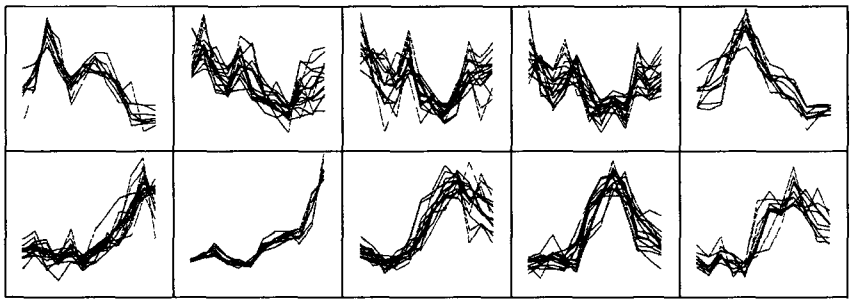


Figure 6.10: Some of the clusters obtained by GenClus over Dataset 3

Figure 6.12 shows a sample output of some clusters of Dataset 2 with genes inserted incrementally. The inserted genes are shown in red color (gray for black & white images) with filled circles at the time points.

2. Cluster Quality

To evaluate the effectiveness of our method as compared to other algorithms, we used two validity measures: z-score and p-value. Next, we report our evaluation of GenClus in terms of these measures.

Table 6.2: z-scores for GenClus and its counterparts for Dataset 2

Method Applied	No. of Clusters	z-score	Total no. of genes
k-means	62	5.57	614
SOM	42	5.78	614
DCCA	42	-0.78	614
RDClust	42	6.61	614
GenClus	61	7.39	614

Table 6.3: z-scores for InGenClus and GenClus for Dataset 1

Method Applied	No. of Clusters	z-score	Total no. of genes
GenClus	21	11.68	384
InGenClus	21	11.68	384

a) z-score

Z-score [GR02] is calculated by investigating the relationship between a clustering result and the functional annotation of the genes in the cluster. We have used Gibbons ClusterJudge [GR02] tool to calculate the z-score. A higher value of z-score indicates that genes would be better clustered by function, indicating a more biologically relevant clustering result. To assess the quality of GenClus, we employed z-score [GR02] as the measure of agreement. To test the performance of the clustering algorithm, we compared the clusters identified by our method with the results from k-means, SOM, DCCA and RDClust. The result of applying GenClus on the reduced form of Dataset 2 is shown in Table 6.2. This table shows that GenClus performed better than the other algorithms in terms of z-score. Similarly, InGenClus was implemented and tested over various datasets. The results were compared with GenClus and have been found satisfactory. Some of the results obtained by InGenClus over Dataset 2 are reported in Figure 6.12. It has been found that InGenClus yields the same result as GenClus, as can be observed from Table 6.3.

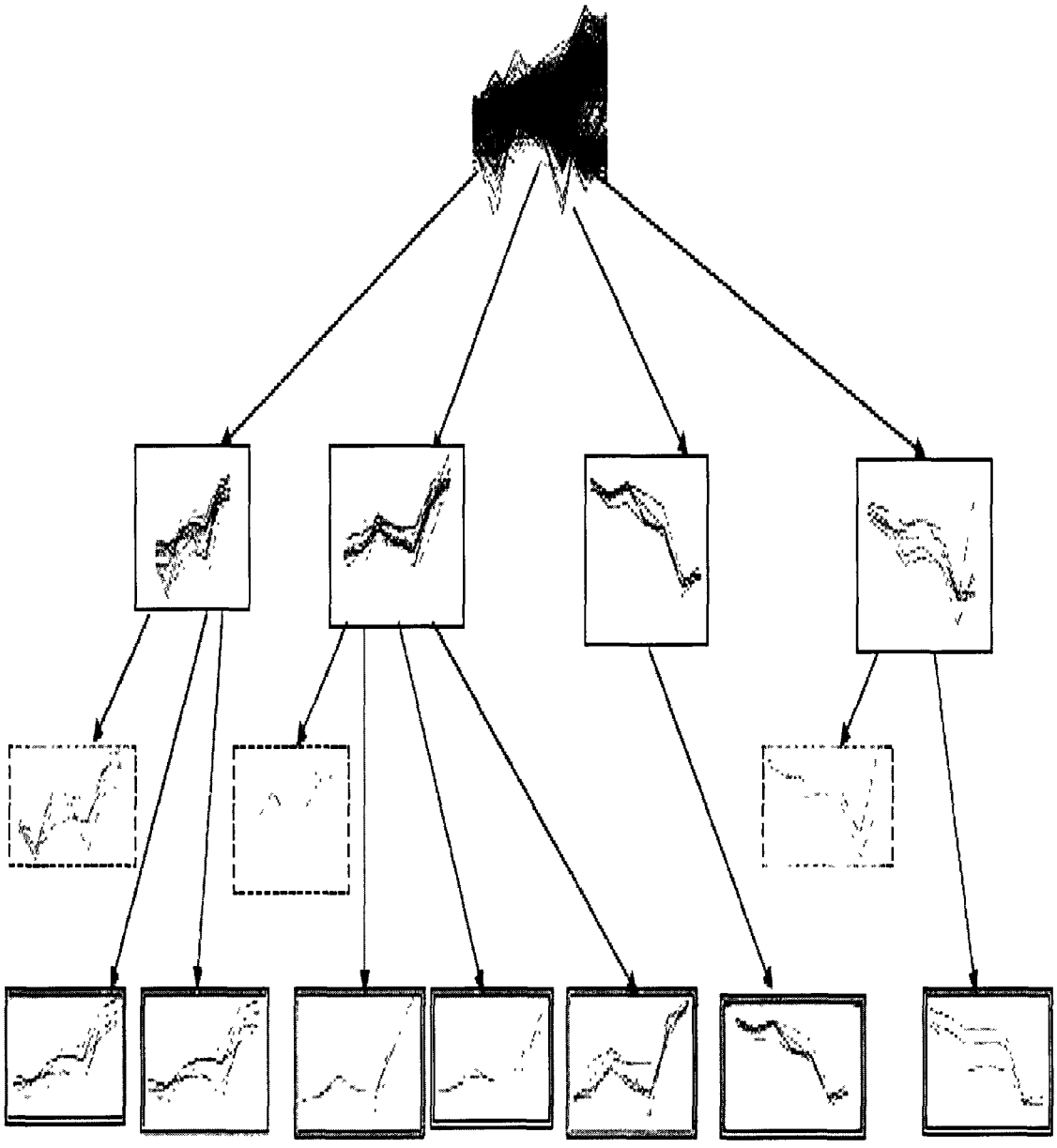


Figure 6.11: Hierarchy of four clusters of Dataset 2.

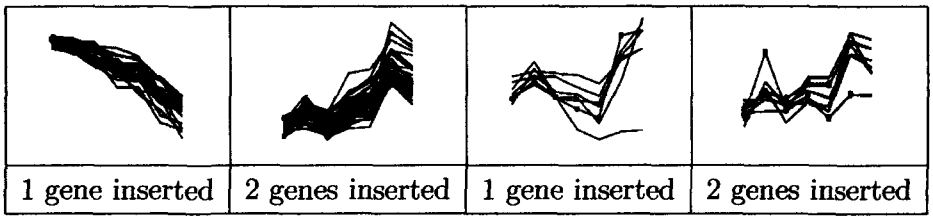


Figure 6.12: Some of the clusters obtained by InGenClus over data incrementally updated from Dataset 2

b) Biological significance

The biological relevance of a cluster can be verified based on the gene ontology (GO) annotation database *http://db.yeastgenome.org/cgi-bin/GO/goTermFinder*. It is used to test the functional enrichment of a group of genes in terms of three structured controlled ontologies, *viz.*, associated biological processes, molecular functions and biological components. The functional enrichment of each GO category in each of the clusters obtained is calculated by its p-value ([THC+99]). The p-value is computed using a cumulative hyper-geometric distribution. It measures the probability of finding the number of genes involved in a given GO term (i.e., function, process, component) within a cluster. The genes in a cluster are evaluated for the statistical significance by computing the p-value for each GO category. This signifies how well the genes in the cluster match with the different GO categories. p-value represents the probability of observing the number of genes from a specific GO functional category within each cluster. A low p-value indicates the genes belonging to the enriched functional categories are biologically significant in the corresponding clusters. To compute the p-value, we used the software FuncAssociate [B+03]. FuncAssociate [B+03] computes the hyper-geometric functional enrichment score based on Molecular Function and Biological Process annotations. The resulting scores are adjusted for multiple hypothesis testing using Monte Carlo simulations. The enriched functional categories for some of the clusters obtained by GenClus method on Dataset 2 are listed in Table 6.4. The functional enrichment of each GO category in each of the clusters is calculated by its p-value. We have reported p-values $< e - 06$. Of the 61 clusters obtained from the dataset, the cluster C6 contains several enriched categories on 'ribosome'. The highly enriched categories in C6 is the 'ribosome' with

Table 6.4: P-value of Dataset 2

Cluster	P-value	GO number	GO category
C1	1e-10	GO 0006119	oxidative phosphorylation
	5 4e-10	GO 0006091	generation of precursor metabolites and energy
	2 8e-08	GO 0022900	electron transport chain
	2 8e-08	GO 0022904	respiratory electron transport chain
	2 8e-08	GO 0042773	ATP synthesis coupled electron transport
	2 8e-08	GO 0042775	organelle ATP synthesis coupled electron transport
	2 8e-08	GO 0055114	oxidation reduction
	7.2e-08	GO 0005739	mitochondrion
	7 9e-08	GO 0044455	mitochondrial membrane part
	1 5e-07	GO 0015078	hydrogen ion transmembrane transporter activity
	1 9e-07	GO 0005743	mitochondrial inner membrane
	2 8e-07	GO 0015077	monovalent inorganic cation transmembrane transporter activity
	3 3e-07	GO 0019866	organelle inner membrane
	3 5e-07	GO 0031966	mitochondrial membrane
	7 6e-07	GO 0005740	mitochondrial envelope
C2	1 5e-11	GO 0042254	ribosome biogenesis and assembly
	4e-11	GO 0005730	nucleolus
	2 8e-10	GO 0022613	ribonucleoprotein complex biogenesis and assembly
	3e-10	GO 0043228	non-membrane-bounded organelle
	3e-10	GO 0043232	intracellular non-membrane-bounded organelle
	4 9e-07	GO 0042273	ribosomal large subunit biogenesis and assembly
	9 4e-07	GO 0006364	rRNA processing
C3	7 7e-10	GO 0042254	ribosome biogenesis and assembly
	8 6e-10	GO 0022613	ribonucleoprotein complex biogenesis and assembly
C5	1 4e-14	GO 0006119	oxidative phosphorylation
	8 5e-14	GO 0044455	mitochondrial membrane part
	6 3e-11	GO 0015078	hydrogen ion transmembrane transporter activity
	9 8e-11	GO 0006091	generation of precursor metabolites and energy
	1 4e-10	GO 0015077	monovalent inorganic cation transmembrane transporter activity
	1 8e-09	GO 0005753	mitochondrial proton-transporting ATP synthase complex
	1 8e-09	GO 0045259	proton-transporting ATP synthase complex
	3 8e-09	GO 0005743	mitochondrial inner membrane
	7 2e-09	GO 0019866	organelle inner membrane
	1e-08	GO 0015985	energy coupled proton transport, down electrochemical gradient
	1e-08	GO 0015986	ATP synthesis coupled proton transport
	1 3e-08	GO 0006754	ATP biosynthetic process

Cluster	P value	GO number	GO category
	1 3e-08	GO 0046034	ATP metabolic process
	2e-08	GO 0022890	inorganic cation transmembrane transporter activity
	2 6e 08	GO 0005746	mitochondrial respiratory chain
	2 6e-08	GO 0009144	purine nucleoside triphosphate metabolic process
	2 6e-08	GO 0009145	purine nucleoside triphosphate biosynthetic process
	2 6e-08	GO 0009205	purine ribonucleoside triphosphate metabolic process
C5	2 6e-08	GO 0009206	purine ribonucleoside triphosphate biosynthetic process
	4e-08	GO 0009199	ribonucleoside triphosphate metabolic process
	4e-08	GO 0009201	ribonucleoside triphosphate biosynthetsic process
	4 4e-08	GO 0016310	phosphorylation
	5 8e-08	GO 0009142	nucleoside triphosphate biosynthetic process
	6 2e-08	GO 0008324	cation transmembrane transporter activity
	7e-08	GO 0016469	proton-transporting two-sector ATPase complex
	7 1e-08	GO 0031966	mitochondrial membrane
	8 3e-08	GO 0009141	nucleoside triphosphate metabolic process
	1 6e-07	GO 0005740	mitochondrial envelope
	1 6e-07	GO 0006818	hydrogen transport
	1 6e-07	GO 0015992	proton transport
	2 7e-07	GO 0015075	ion transmembrane transporter activity
	6 4e-07	GO 0022900	electron transport chain
	6 4e-07	GO 0022904	respiratory electron transport chain
	6 4e-07	GO 0042773	ATP synthesis coupled electron transport
	6 4e-07	GO 0042775	organelle ATP synthesis coupled electron transport
	6 4e-07	GO 0055114	oxidation reduction
	6 6e-07	GO 0006793	phosphorous metabolic process
	6 6e-07	GO 0006796	phosphate metabolic process
	8 1e-07	GO 0009152	purine ribonucleotide biosynthetic process
	9 1e-07	GO 0009150	purine ribonucleotide metabolic process
C6	2 7e-14	GO 0043228	non-membrane-bounded organelle
	2 7e-14	GO 0043232	intracellular non-membrane-bounded organelle
	3 6e-13	GO 0005840	ribosome
	1 1e-12	GO 0030529	ribonucleoprotein complex
	1 5e-12	GO 0042254	ribosome biogenesis and assembly
	1 7e-12	GO 0022613	ribonucleoprotein complex biogenesis and assembly
	2 2e-12	GO 0022626	cytosolic ribosome
	9 5e-12	GO 0003735	structural constituent of ribosome
	3e-11	GO 0044445	cytosolic part
	1 7e-10	GO 0033279	ribosomal subunit

Cluster	P-value	GO number	GO category
	3.7e-09	GO:0005198	structural molecule activity
	2.6e-08	GO:0044249	cellular biosynthetic process
C6	3.1e-08	GO:0022625	cytosolic large ribosomal subunit
	4.4e-08	GO:0005730	nucleolus
	3.4e-07	GO:0006412	translation
	6.8e-07	GO:0015934	large ribosomal subunit
C7	2.5e-09	GO:0005991	trehalose metabolic process
	1.9e-08	GO:0005975	carbohydrate metabolic process
	2.6e-08	GO:0044262	cellular carbohydrate metabolic process
	5.8e-08	GO:0005992	trehalose biosynthetic process
	5.8e-08	GO:0046351	disaccharide biosynthetic process
	1.9e-07	GO:0046164	alcohol catabolic process
	9.4e-07	GO:0005996	monosaccharide metabolic process

a p-value of 3.6e-13. The GO category 'ribonucleoprotein complex' is also highly enriched in this cluster with p-value of 1.1e-12. Cluster C6 also contains the highly enriched categories on 'non-membrane-bounded organelle' with a p-value of 2.7e-14. Cluster C5 contains an enriched category 'oxidative phosphorylation' with p-value of 1.4e-14. C5 also contains several enriched categories on 'mitochondria'. Cluster C2 contains several enriched categories on 'biogenesis'. The highly enriched categories in C2 are the 'ribosome biogenesis and assembly' with p-value of 1.5e-11, 'ribonucleoprotein complex biogenesis and assembly' with p-value of 2.8e-10 and 'ribosomal large subunit biogenesis and assembly' with p-value of 4.9e-07. In the cluster C7 all the functionally enriched categories are from Biological Process annotation with 'trehalose metabolic process' with a p-value of 2.5e-09 being the highly enriched one. From the Table 6.4, we can conclude that GenClus shows a good enrichment of functional categories.

In this section, we have presented GenClus which can detect clusters over gene expression data without the use of any proximity measures. The clusters obtained were found satisfactory when compared with other relevant algorithms. An incremental version of GenClus is also presented for handling incremental data. Gene expression data have highly intersecting clusters, the detection of which is very difficult. Hierarchical algorithms helps in identifying the clusters at different levels. In the next section we present a hierarchical density based technique for clustering gene expression data which can also identify clusters in the subspaces of the data.

6.6 GeneClusTree

GeneClusTree is a gene based clustering technique which attempts to cluster the gene dataset using a tree-based density approach. GeneClusTree does not use any proximity measure during clustering the genes and is therefore free from the restriction offered by them. The other two important advantages of GeneClusTree are:

- capable of handling noisy datasets;
- does not require the number of clusters apriori.

6.6.1 Basics of GeneClusTree

At first, the gene expression data is normalized to have mean 0 and standard deviation 1. Expression data having a low variance across conditions as well as data having more than 3-fold variation are filtered out. Discretization is then performed on this normalized expression data by retaining the up- or down- regulation information in each of the conditions for a particular gene as discussed in Section 6.5.1. After the regulation based discretization process, each gene will now have a regulation pattern ($\wp$) of 0, 1, and -1 across the conditions or time points and is represented as a string.

To avoid the restrictions caused due to the use of any proximity measure, GeneClusTree exploits the angle information computed over the normalized expression values. The angle information is computed condition-wise for each of the gene profiles based on their normalized expression values. The angle information gives the trend angle between each pair of conditions. We illustrate the whole gene-condition space as a graph with conditions across x-axis and expression levels along y-axis. Let the x-axis for a gene g_i be denoted as x_{t_j} for condition t_j and its corresponding y-axis be denoted by the expression value ε_{g_i,t_j}. Now, for the gene g_i, the angle information for each of its T conditions is computed according to the formula[1] given in Equation below.

$$\alpha_{\varepsilon_{g_i,t_j},\varepsilon_{g_i,t_{j+1}}} = arccos\left(\frac{\varepsilon_{g_i,t_{j+1}} - \varepsilon_{g_i,t_j}}{\sqrt{(\varepsilon_{g_i,t_{j+1}} - \varepsilon_{g_i,t_j})^2 + (x_{t_{j+1}} - x_{t_j})^2}} \right)$$

[1] Available in http://mathematics.learnhub.com/lesson/5945-trigonometry-basics

Each gene g_i will now have a pattern of angle information α_{g_i} consisting of T values. This angle information is then further discretized by dividing it into discrete equal intervals depending on their angle values where the width of the interval is a user input. After discretization of the angle values, each gene, g_i, will have a pattern of *angle_ids* (α'_{g_i}) across conditions, the *angle_id* value at the k^{th} condition is denoted as α'_{g_i,t_k}. The regulation information and discretized angle patterns are used together to cluster the gene expression dataset using a tree-based density approach. A string matching approach is used for matching the regulation and the discretized angle patterns of two genes. Next, we give some definitions which provide the foundation of GeneClusTree.

Definition 29. Matched Subspace: Let $g_i, g_j \in G$ and $\wp(g_i), \wp(g_j)$ denote their corresponding regulation patterns. Then the matched subspace, $M(g_i, g_j)$, of (g_i, g_j) is the set of $\hbar$ conditions where both g_i and g_j match and $\theta^* \leq \hbar \leq T$ and θ^* is a user defined parameter.

Definition 30. Maximal Matched Subspace: A matched subspace can be defined as maximal if it is a matched subspace and no superset of this can be found to be a matched subspace, or,
A pair of genes (g_i, g_j) is said to be maximally matched if the cardinality of the subset of conditions over which they are matched is maximal i.e., over no superset of conditions g_i and g_j are matched.

Definition 31. Neighbor of a gene: A gene g_j is said to be a neighbor of gene g_i i.e., $g_j \in N_{level}(g_i)$, iff

i. g_i and g_j are in the same level, say, *level*,

ii. $\mid M(g_i, g_j) \mid \geq \theta^*$, and

iii. $\alpha'_{g_j,t_\hbar} = [\alpha'_{g_i,t_\hbar} + \delta', \alpha'_{g_i,t_\hbar} - \delta']$, where $t_\hbar$ refers to $\hbar$ conditions and δ' has an initial value of 1 in the first level for each sub-tree and is incremented by 1 in every subsequent levels.

For regulation matching, GeneClusTree initially attempts to find neighbors of a gene g_i over full set of conditions i.e., T number of conditions. If no match is found,

the number of conditions is decremented by 1 at each step upto a certain threshold (say, θ^*) till a match occurs i.e., $\wp(g_i) \approx \wp(g_j)$. However, at each subsequent step, the previously computed matching information is used which makes the searching more efficient.

Definition 32. Initiator: A gene g_i in the level say, *level*, is said to be an initiator if $\mid N_{level}(g_i) \mid \geq \sigma$.

The neighborhood of a gene g_i is searched for genes satisfying the *initiator* condition. If no neighbor gene is found, then the process is repeated with another unclassified gene. In our experiments we have obtained good results for $\sigma = 2$.

Definition 33. Node: A node n_i in the level say, *level*, is a non-empty subset of genes of G where, any gene $g_j \in n_i$ is either
(i) itself an initiator gene, or
(ii) is within the neighborhood of an initiator gene $g_i \in n_i$ i.e., $g_j \in N_{level}(g_i)$.

Definition 34. Node Reference Vector: Reference Vector of a node is the subset of conditions where all the genes belonging to that node match maximally.

The Reference Vector of a node n_i (RV_{n_i}) to which, say, genes g_i and g_j belong is computed as follows:

$$RV_{n_i} = \begin{cases} 1 & \text{if } \wp_{g_i,t} = \wp_{g_j,t} = 1 \\ 0 & \text{if } \wp_{g_i,t} = \wp_{g_j,t} = 0 \\ -1 & \text{if } \wp_{g_i,t} = \wp_{g_j,t} = -1 \\ x & \text{otherwise} \end{cases}$$

Definition 35. Intra-node reachability: A pair of genes (g_i, g_j) in any level, say *level*, is said to be intra-node reachable if,
(i) one of them is an initiator and the other is a neighbor of it, or
(ii) another gene g_k is an initiator and $g_i, g_j \in N_{level}(g_k)$, or
(iii) both g_i, g_j are initiators and they are neighbors to each other i.e., either $g_i \in N_{level}(g_j)$ or $g_j \in N_{level}(g_i)$.

The intra-node reachable genes satisfy the condition that they match in the same subset of conditions of regulation pattern.

Definition 36. Inter-node Reachability: A gene $g_j \in n_c$ is said to be inter-node reachable from another gene $g_i \in n_p$, (where n_p is the parent of n_c), if $\wp_{g_j}$ matches with $\wp_{g_i}$ in a total of $(T - (l_c - 1))$ number of conditions, where, l_c is the level of n_c.

Finding subspace clusters in different levels gives different level of finer clustering of a dataset which may be useful for the biologists.

Definition 37. Maximal-Space cluster: A node n_i is said to be maximal-space cluster if n_i is created at the first level (i.e., level 1) and the set of genes in n_i match over a set of $\hbar$ conditions, where $\theta^* \leq \hbar \leq T$.

Definition 38. Reduced-Space cluster: A node n_i is a said to be a reduced-space cluster if n_i is created in the j^{th} level and the set of genes in n_i match over a set of $(\hbar - (j - 1))$ conditions where $\hbar$ is cardinality of the set of conditions in which the genes in the parent node of n_i match in the $(j-1)^{th}$ level where $2 \leq j \leq (\hbar - (\theta^* - 1))$.

In the rest of the chapter, we will use the terms *node* and *subspace cluster* interchangeably to represent a cluster over a subset of conditions.

Definition 39. Noise Genes: Let $n_1, n_2, \cdots n_p$ be the set of subspace clusters of G, then noise genes in G is the set of genes not belonging to any subspace cluster n_i, *i.e.*,

$$noise = \{g_x \in G \mid \forall i : g_x \notin n_i\}$$

GeneClusTree starts by creating a tree structure in a depth-first manner with an empty node as the root. The root is at level 0 and is connected to all the nodes in level 1. The nodes in level 1 are created by a density based approach and each of these nodes is the basis of formation of the reduced-space clusters of the sub-tree. The process of creating a level 1 node i.e., a maximal space cluster starts with an arbitrary unclassified gene g_i and the neighborhood of g_i is searched to check whether it is an initiator. If no gene is found to satisfy the neighborhood condition with g_i, then the process restarts with another unclassified gene. On the other hand, if g_i is an initiator gene it initiates the process of creating a new node with a node reference vector formed according to Definition 34. Then the process proceeds with finding all

the genes that satisfies the intra-node reachability condition with g_i in terms of the node reference vector and are assigned to the same node to which g_i belongs. If any gene g_j from the set of intra-node reachable genes of g_i satisfies the *initiator* gene condition, then the node expansion proceeds with the gene g_j. The process continues till no more genes can be assigned to the node.

Each of the nodes in level 1 is a maximal-space cluster and determines the nodes to be formed as reduced-space clusters, across different subset of conditions, in the next level of the sub-tree. After completion of the formation of the node(s) of a particular level in a sub-tree, the value of level is incremented by 1.

Each of the nodes formed at level 1 becomes the parent node of the sub-trees formed at the next level (i.e., level 2). Similarly, for the nodes in level i, their parents will be in level (i - 1). Also, with the increase in the height of the tree, the cardinality of the matched condition set decreases from parent to child by 1 at each level. For a particular sub-tree, the genes in each of the i^{th} nodes agrees over a set of $(\hbar - (level - 1))$ conditions of the parent node's reference vector. Genes belonging to the sibling node(s) at the same level in a particular sub-tree have the same cardinality of matched conditions, however the match is over different set(s) of conditions.

On completion of the child nodes along with their sibling nodes of a particular node in a sub-tree, the process continues similarly in the next level until the sub-tree reaches a depth of θ^* or no more nodes can be added to the sub-tree. The process then backtracks to level 1 and finds the next maximal subspace cluster and inserts it as a child of the root. The sub-tree of this node is created in the similar manner as described before. The whole process repeats itself until no more maximal subspace clusters are inserted in level 1 of the tree. All the remaining unclassified genes are treated as noise genes. The algorithm is given in detail in Figures 6.13(a), (b) and 6.14.

The following lemmas are formulated from the definitions of GeneClusTree.

Lemma 7. A gene g_i belonging to $n_{1,i}$ ($n_{1,i}$ is the i^{th} subspace cluster of level 1) cannot be a neighbor of any gene $g_j \in n_{1,j}$, where $n_{1,j}$ is the j^{th} subspace cluster of

Tree_creation(D_G, node_id)

$l = level$

FOR all $g_i \in D_G$ do

IF g_i.classified != 1 and chk_ini_condition(g_i) == true then

create_node(g_i.no, p_id, temp, $temp_{count}$, node_id, l);

WHILE((T - (l - 1)) $\geq \theta^*$) do

l++;

FOR all $g_i \in D_G$ do

IF g_i.classified != 1 then

p_id = chk_gene_parent(g_i.no, l);

IF p_id > -1 and chk_ini_condition(g_i) == true then

create_node(g_i.no, p_id, temp, $temp_{count}$, node_id, l);

End IF

End IF

End FOR

End WHILE

l = 1;

End IF

End FOR

(a) Algorithm for Tree Creation

create_node(no, p_id, temp, $temp_{count}$, id, l)

$node_{id}$ = new node();

$node_{id}$.temp = temp;

$node_{id}.templ_{count} = temp_{count}$;

$node_{id}$.p_node = p_id;

$node_{id}$.core_gene = no;

$node_{id}$.id = id;

$node_{id}$.level = l;

expand_node(no, id, $node_{id}$.temp, $temp_{count}$, l);

temp = NULL;

$temp_{count} = 0$;

node_id++;

Figure 6.13: (b) Algorithm for Node creation

expand_node(no, id, temp, $temp_{count}$, l)

IF g_{no}.classified == 1 then
  Return;
Else
  g_{no}.classified = 1;
  g_{no}.node_id = id;
  FOR all $g_i \in D_G$ do
    IF g_i.classified != 1 then
      $match_{count}$ = find_match_temp(g_i, temp, $temp_{count}$);
      IF $match_{count}$ >= ($temp_{count}$ - (l - 1)) and $match_{count}$ >= θ^* then
        expand_node(g_i.no, id, temp, $temp_{count}$, l);
      End IF
    End IF
  End FOR
End IF

Figure 6.14: Algorithm for Node expansion

level 1.

Proof. Suppose, $g_i \in n_{1,i}$ and $g_j \in n_{1,j}$ and let $g_i \in N_{level}(g_j)$. Then, g_i is intra-node reachable from g_j according to Definition 35. Therefore, both g_i and g_j belongs to the same node (subspace cluster). Therefore, we come to a contradiction and hence the proof. □

Lemma 8. A gene g_j belonging to $n_{i,j}$ may not belong to $n_{(i-1),k}$ (i = 2, 3,$\cdots$, θ) where $n_{i,j}$ refers to j^{th} subspace cluster of level i.

Proof. Let $g_j \in n_{i,j}$, then according to Definition 36, g_j is inter node reachable from any node $n_{i-1,k}$ in level $(i-1)$ and $g_k \in n_{i-1,k}$. Then $\wp(g_j)$ matches in a total of $(T-((i-1)-1))$ conditions, i.e., one condition less than g_k. The reference vector of $n_{i,j}$ will be same as that for $n_{i-1,k}$ except for the condition where g_k and g_j do not match. Therefore, $g_j \notin n_{i-1,k}$ and hence the proof. □

Lemma 9. Genes $g_i, g_j \in G$. Now, if $g_i \in n_i$ and $g_j \in n_j$ where n_i and n_j are two subspace clusters, then g_i and g_j are not intra-node reachable.

Proof. Let g_i and g_j be two intra-node reachable genes and $g_i \in n_i$ and $g_j \in n_j$, where n_i and n_j are two subspace clusters at any level. Then, according to Definition 35, either they are neighbors or both of them are neighbors of another initiator gene, that is, g_i and g_j must be in the same subspace cluster according to Definition 33. Therefore, we come to a contradiction and hence the proof. □

Lemma 10. Assume genes $g_i, g_j \in G$ and $g_i \in n_i$ and $g_j \in n_j$ where n_i and n_j are two subspace clusters (n_i and n_j are not parent-child or vice versa), then g_i and g_j are not inter-node reachable.

Proof. Let g_i and g_j be two inter-node reachable genes and $g_i \in n_i$ and $g_j \in n_j$, where n_i and n_j are two subspace clusters but do not share a parent-child relationship between them. However, according to Definition 36, two genes are inter-node reachable if one of them belongs to a parent node and the other to its child. Therefore, n_i and n_j should be parent-child or vice versa. Thus, we come to a contradiction and hence the proof. □

Theorem 1. Two genes g_i and g_j belonging to two different nodes are not coherent.

Proof. Let $g_i \in n_i$ and $g_j \in n_j$. Now, as per *lemma* 1, g_i cannot be a neighbor of g_j. Again, since genes g_i and g_j are not neighbors, then the conditions given in Definition 30 do not satisfy and hence they are not coherent. □

Theorem 2. A gene g_i without a neighbor is a noise.

Proof. A gene g_i without a neighbor is neither intra-node nor inter-node reachable from any other node, hence such a gene g_i can be trivially proved to be a noise gene according to 39, *lemma* 3 and *lemma* 4. □

6.6.2 Performance Evaluation

GeneClusTree was implemented in Java in Windows environment and evaluated with several real-life datasets. Of the various datasets, the results of some of the datasets as given in Table 6.1 are reported in this chapter. All the datasets are normalized to have mean 0 and standard deviation 1. The datasets have been obtained from *http://faculty.washington.edu /kayee/cluster.*

1. Results

Figure 6.15 shows some of the maximal space and reduced space clusters of Dataset 1. The clusters formed from the full form of Dataset 2 are shown in Figure 6.16. Figure 6.17 shows some of the clusters obtained from the reduced form of Dataset 2. The maximal and reduced space clusters of Dataset 3 are shown in Figure 6.18.

2. Cluster Quality

In order to validate our clustering result, we employ z-score [GR02] as the measure of agreement. The biological relevance of a cluster can be verified based on the gene ontology (GO) annotation database *http://db.yeastgenome.org/cgi-bin/GO/goTerm Finder.* It is used to test the functional enrichment of a group of genes in terms of three structured ontologies, *viz.*, associated biological processes, molecular functions and biological components. The functional enrichment of each GO category in each

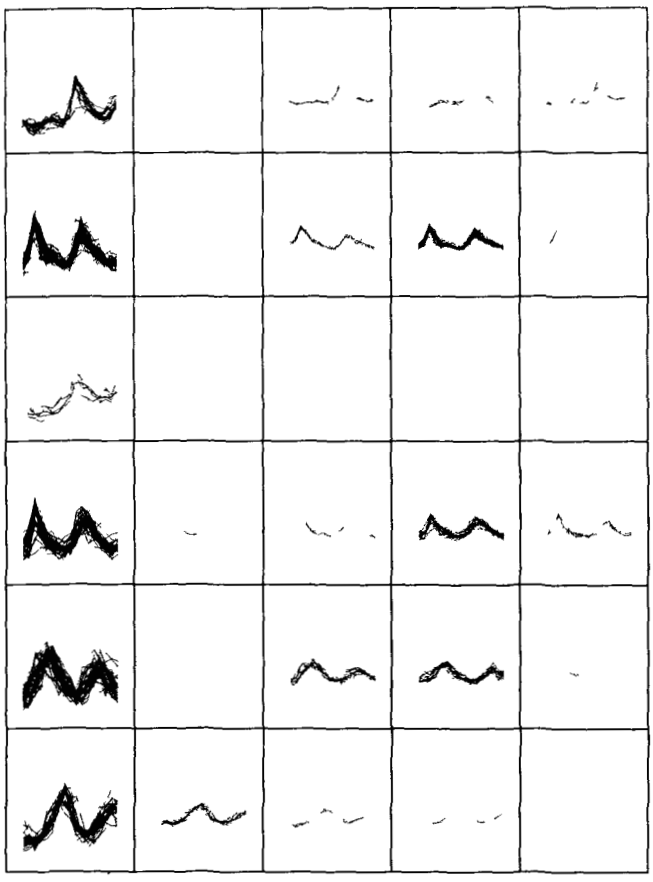


Figure 6.15: Each of the rows represents the six clusters formed from Dataset 1. Starting from the second column of each row, the reduced space clusters are illustrated for the maximal space cluster given in the first column.

of the clusters obtained is calculated by its *p-value.*

As given in section 6.5.3 2a), to assess the quality of GeneClusTree, we employed z-score [GR02] as the measure of agreement. Higher the value of z, better the cluster results indicating more biologically relevant clusters of genes. z-score is calculated by *investigating the relation between a clustering result and the functional annotation* of the genes in the cluster. We have used Gibbons ClusterJudge [GR02] tool to calculate the z-score. To test the performance of the clustering algorithm, we compared clusters identified by GeneClusTree with the 'ground truth' and with the results from RDClust, DCCA and UPGMA. The result of applying the z-score on Dataset 1 is

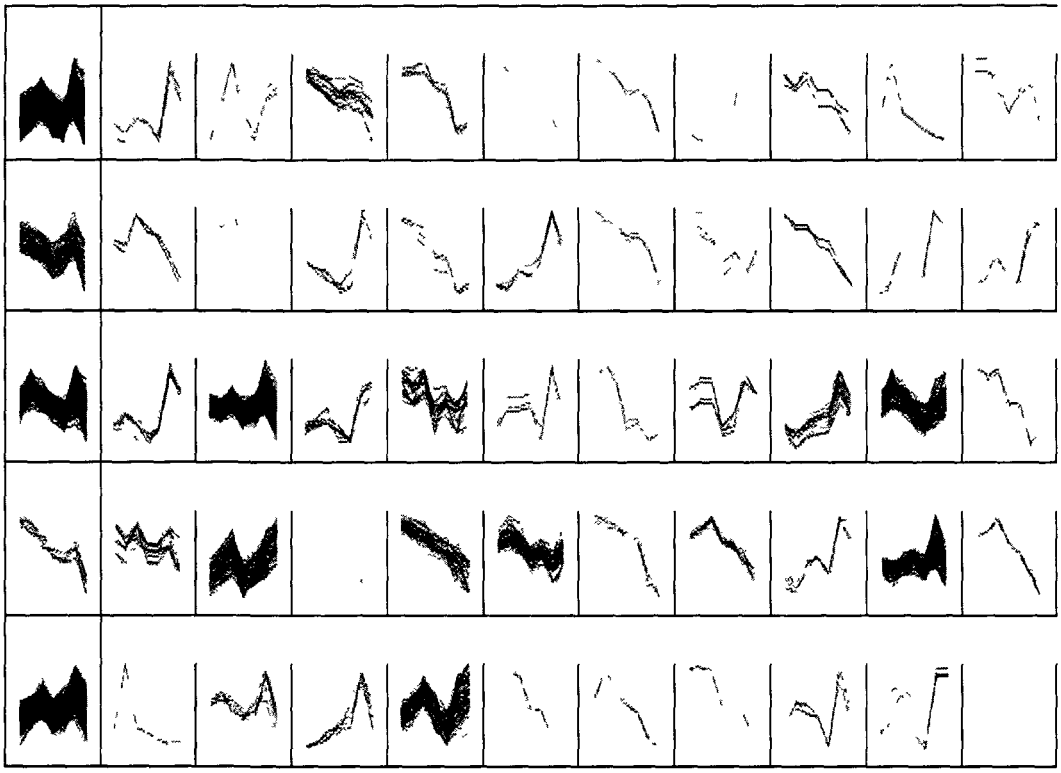


Figure 6.16: Some of the clusters from the Dataset 2

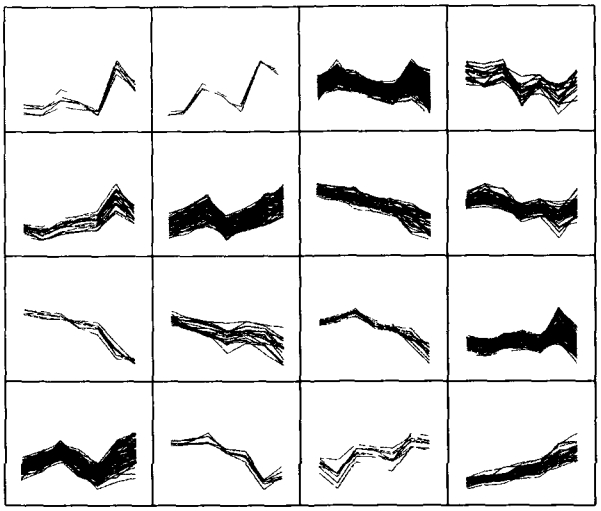


Figure 6.17: Some of the clusters obtained from the reduced form of Dataset 2.

Table 6.5: z-scores for GeneClusTree and its counterparts for Dataset 1

Method Applied	No. of Clusters	z-score	Total no. of genes
UPGMA	16	5.57	384
DCCA	10	6.2	384
RDClust	10	6.95	384
GeneClusTree	17	7.42	384

shown in Table 6.5. In this table, the proposed algorithm is compared with the well known agglomerative hierarchical algorithm, UPGMA, DCCA and RDCLust. Table 6.5 clearly shows that GeneClusTree outperforms UPGMA, RDClust and DCCA w.r.t. the cluster quality. We note here that unlike k-means our method does not require the number of clusters as an input parameter. It detects the clusters present in the dataset automatically and gives the rest as noise. However, the algorithm UPGMA requires the input parameter $cutoff$.

As given in section 6.5.3 2b), the functional enrichment of each GO category in each of the clusters obtained is calculated by its *p-value* ([THC+99]). A low p-value indicates the genes belonging to the enriched functional categories are biologically significant in the corresponding clusters. To compute the p-value, we used the software

FuncAssociate [B+03]. To restrict the size of this chapter, the enriched functional categories for only three clusters obtained by GeneClusTree on Dataset 1 are partially listed in Table 6.6. The functional enrichment of each GO category in each of the clusters is calculated by its p-value. Cluster C2 contains genes involved in DNA replication with the highly enriched category being 'MCM complex' with a p-value of 1.1×10^{-12}. The highly enriched categories in C5 is the 'cellular bud' with a p-value of 7.0×10^{-07}. The genes in cluster C10 are involved in cell cycle. C10 contains the highly enriched cellular components of 'DNA metabolic process', 'DNA replication', 'chromosome', 'chromosomal part', 'cell cycle, etc. with p-values of 1.8×10^{-22}, 1.8×10^{-21}, 9.7×10^{-21}, 1.5×10^{-20} and 8.8×10^{-19} being the highly enriched one. From the Table 6.6, we can conclude that GeneClusTree shows a good enrichment of functional categories and therefore project a good biological significance.

We present an effective tree-based clustering technique (GeneClusTree) for finding clusters over gene expression data. GeneClusTree attempts to find all the clusters over subspaces using a tree-based density approach by scanning the whole database in minimum possible scans of the dataset. Another important advantage of GeneClusTree is that it is free from the restrictions of using a proximity measure. Our algorithm works by finding the maximal space clusters and then proceeds in finding the reduced space clusters. The clusters are represented as a tree with the reduced space clusters as the child of its respective maximal space cluster. Effectiveness of GeneClusTree is established in terms of well known z-score measure and p-value over several real-life datasets. Using z-score analysis we show that GeneClusTree outperforms other comparable algorithms. The p-value analysis shows that our technique is capable in detecting biologically relevant clusters from gene expression data.

6.7 Discussion

This work presents a tree-based density approach which finds useful subgroups of genes within a cluster and obtains a tree structure of the dataset where the clusters at the bottom level gives the finer clustering of the dataset. GeneClusTree does not require the number of clusters apriori and the clusters obtained have been found

Table 6.6: *P*-value of some of the clusters of Dataset 1

Cluster	P-value	GO number	GO category
C2	1 1e-12	GO 0042555	MCM complex
	5 2e-10	GO 0005656	pre-replicative complex
	5 2e-10	GO 0006267	pre-replicative complex assembly
	1 5e-09	GO 0000084	S phase of mitotic cell cycle
	3 5e-09	GO 0031261	DNA replication preinitiation complex
	3 5e-09	GO 0043596	nuclear replication fork
	3 5e-09	GO 0005739	S phase
	1 1e-08	GO 0044455	DNA replication origin binding
	3 8e-08	GO 0051329	interfiphase of mitotic cell cycle
	4 1e-08	GO 0051325	interphase
	5 4e-08	GO 0005657	replication fork
	6 3e-08	GO 0006270	DNA replication initiation
	7 2e-08	GO 0008094	DNA dependent ATPase activity
	1 1e-07	GO 0009378	four-way junction helicase activity
	1 2e-07	GO 0022403	cell cycle phase
	3 9e-07	GO 0022402	cell cycle process
C5	7e-07	GO 0005933	cellular bud
	5 1e-06	GO 0004857	enzyme inhibitor activity
	8 3e-06	GO 0004860	protein kinase inhibitor activity
	8 3e-06	GO 0019210	kinase inhibitor activity
	9 6e-06	GO 0030427	site of polarized growth
	2 8e-05	GO 0005935	cellular bud neck
	4 7e-05	GO 0019887	protein kinase regulator activity
	5 6e-05	GO 0019207	kinase regulator activity
	6 7e-05	GO 0004861	cyclin-dependent protein kinase inhibitor activity
C10	1 8e-22	GO 0006259	DNA metabolic process
	1 8e-21	GO 0006260	DNA replication
	9 7e-21	GO 0005694	chromosome
	1 5e-20	GO 0044427	chromosomal part
	8 8e-19	GO 0007049	cell cycle
	1 4e-18	GO 0006281	DNA repair
	1 3e-16	GO 0006974	response to DNA damage stimulus
	4 7e-16	GO 0009719	response to endogenous stimulus
	6 4e-16	GO 0006261	DNA-dependent DNA replication
	1e-05	GO 0006261	DNA-dependent DNA replication
	5 6e-14	GO 0022402	cell cycle process
	9 6e-14	GO 0005634	nucleus
	1e-13	GO 0007064	mitotic sister chromatid cohesion
	7 2e-13	GO 0005657	replication fork
	9 1e-13	GO 0022403	cell cycle phase
	3e-12	GO 0051276	chromosome organization and biogenesis
	3 5e-12	GO 0000228	nuclear chromosome
	4 4e-12	GO 0000278	mitotic cell cycle
	6 4e-12	GO 0007062	sister chromatid cohesion
	1 2e-11	GO 0044454	nuclear chromosome part
	7 5e-11	GO 0006273	lagging strand elongation

Cluster	P-value	GO number	GO category
C10	3 1e-10	GO 0043228	non-membrane-bounded organelle
	3 1e-10	GO 0043232	intracellular non-membrane-bounded organelle
	4 2e-10	GO 0006271	DNA strand elongation during DNA replication
	4 2e-10	GO 0022616	DNA strand elongation
	5 1e-10	GO 0030894	replisome
	5 1e-10	GO 0043601	nuclear replisome
	5 7e-10	GO 0006950	response to stress
	9 1e-10	GO 0051052	regulation of DNA metabolic process
	1 1e-09	GO 0000819	sister chromatid segregation
	1 5e-09	GO 0006139	nucleobase, nucleoside, nucleotide and nucleic acid metabolic process
	6 5e-09	GO 0045934	negative regulation of nucleobase, nucleoside, nucleotide and nucleic acid metabolic process
	6 5e-09	GO 0000070	mitotic sister chromatid segregation
	8 6e-09	GO 0043596	nuclear replication fork
	3 1e-08	GO 0051301	cell division
	3 8e-08	GO 0000793	condensed chromosome
	5 4e-08	GO 0031324	negative regulation of cellular metabolic process
	5 4e-08	GO 0003677	DNA binding
	5 7e-08	GO 0009892	negative regulation of metabolic process
	7 8e-08	GO 0000279	M phase

satisfactory on visual inspection and also based on z-score as well as p-values for three real datasets. However, work is going on for establishing the effectiveness of GeneClusTree over more real-life datasets. Also we are trying to incorporate GO term information during node expansion to make the method more effective biologically.

The current information explosion, fueled by the availability of World Wide Web and the huge amount of microarray experiments conducted has led to the ever-increasing volume of data. There is therefore a need to introduce incremental clustering so that updates can be clustered in an incremental manner. As a future direction of our work, we are focusing on introducing an incremental version of GeneClusTree in the line of work of [DBK09b] which would be able to handle datasets that are updated incrementally.

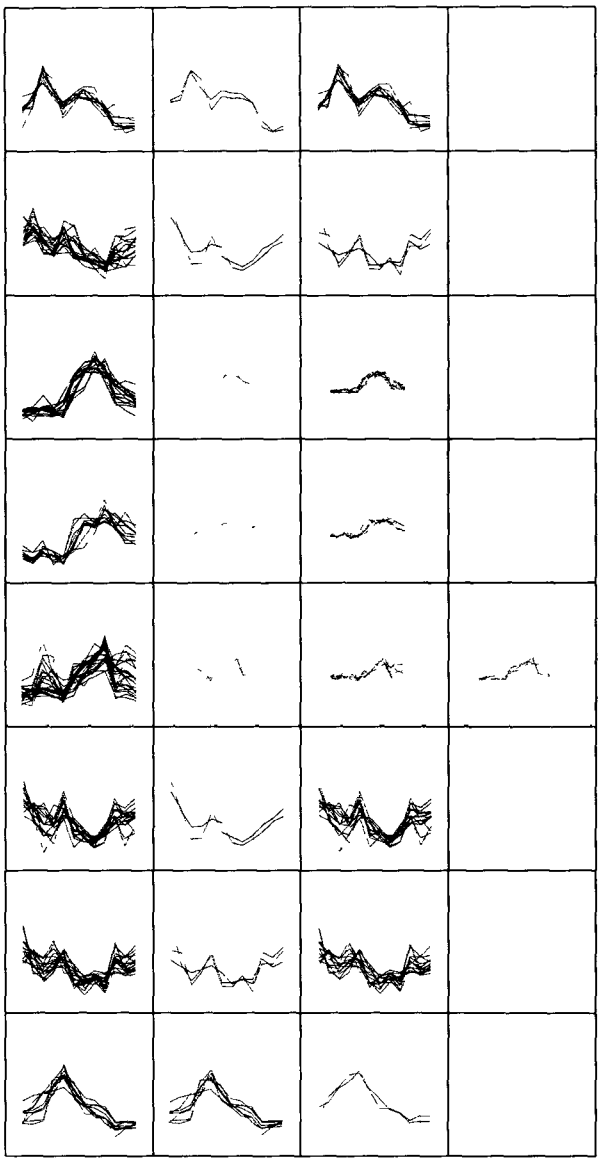


Figure 6.18: Each of the rows represents some of the clusters formed from Dataset 3. Starting from the second column of each row, the reduced space clusters are illustrated for the maximal space cluster given in the first column.

Chapter 7

Conclusions and Future Works

We now conclude this thesis and provide some directions for future scope of work.

7.1 Conclusions

In this thesis, we have developed clustering techniques for three different types of data domains namely 2D spatial data, satellite image data and gene expression data. Detecting global as well as embedded clusters in presence of noise in less time is an important goal in cluster analysis. In our quest to achieve this goal, we have presented GDCT, a fast grid-density based clustering technique for identifying arbitrary shaped clusters of variable density. We have also incorporated an outlier detection module in this algorithm. Identifying clusters from high resolution satellite images has received focus in recent years. This thesis also presents two clustering techniques, SATCLUS and GDSDC, for high-resolution satellite images. Both techniques first find coarse clusters and then reassigns border points to the most appropriate cluster which may have been misclassified during the first step and thus improve the quality of clustering. Satellite images usually have the problem of mixed pixels and handling of such pixel is very important. GDSDC helps in the detection of mixed pixels by including a fuzzy set based approach. Both SATCLUS and GDSDC are better than other comparable algorithms in terms of β measure.

Due to the huge amount of data generated with tremendous progress in data ac-

cumulation techniques, mining useful information from such voluminous data has become a challenge. Parallel and distributed techniques have been used widely for mining such large amount data. In this thesis, we have presented two distributed clustering techniques for spatial data. The first, DGDCT, has been used for clustering massive 2D spatial data and the second, DisClus, has been used for satellite data. Both techniques detect clusters of good quality and the results obtained using synthetic and satellite image datasets establish that the techniques are efficient and obtain scale up.

This thesis also presents two clustering techniques for finding coherent genes from gene expression data. The first technique GenClus identifies useful subgroups of highly coherent genes within a cluster and obtains a hierarchical structure where the sub-clusters give the finer clustering of the dataset. An incremental version of GenClus, i.e., InGenClus is also presented. The second technique is a tree-based density approach which finds useful subgroups of genes within a cluster and obtains a tree structure of the dataset where the clusters at the bottom level give the finer clustering of the dataset. Both GenClus and GeneClusTree do not require the number of clusters apriori and the clusters have been validated based on z-score and p-value measures.

7.2 Future Works

The work reported in this thesis can be expanded and improved in many different ways. Below, we briefly outline future scope of work.

- In SATCLUS, we use a grid-density based approach with a partitioning method to obtain the final clusters. However, satellite images have inherent vagueness in pixel information due to the fact that a single pixel represents quite a lot of data on ground owing to the resolution of the camera used. Therefore, a pixel may belong to more than one cluster. Fuzzy rough set theory exploits the fact that an element can belong to several "soft similarity classes" at the same time with some degree of certainty and therefore provides efficient algorithms for finding hidden patterns in the data. SATCLUS can be further enhanced

by incorporating a rough-fuzzy set theoretic approach to provide an efficient classification scheme.

- SATCLUS and GDSDC have been used with multi-spectral high resolution satellite images. Therefore, there are scopes to extend them to handle hyper-spectral high resolution satellite data.

- In GDSDC, we use a fuzzy membership function to handle the mixed pixels problem present in the border regions of the clusters. As a future direction of work, a sub-pixel approach may be incorporated to handle the mixed pixels.

- For the gene pattern identification, we use two techniques for identifying co-expressed genes. But genes determined to be co-expressed using clustering may not necessarily be co-regulated and hence may not have similar functions. A possible approach may be that the annotated subset of differentially expressed genes be clustered together based on functional similarity and superimposed on top of the clustering techniques to obtain more biologically relevant clusters. In future work, we plan to integrate the analysis of gene expression datasets with biological information regarding functions of genes to identify the co-regulated genes.

Bibliography

[AAS+09] Ram A., Sharma A., Jalall A. S., Singh R., and Agrawal A. EDBSCAN: Enhanced density based spatial clustering of applications with noise. In *Proceedings of IEEE International Advance Computing Conference (IACC 2009)*, Patiala, India, 2009.

[ABKS99] M. Ankerst, M. M. Breuing, H. P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of ACM-SIGMOD 99*, pages 49–60, 1999.

[ABN+99] U. Alon, N. Barkai, Notterman, D. A., K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array. In *Proceedings of National Academy of Sciences*, volume 96(12), pages 6745–6750, USA, 1999.

[ACN09] M. Awad, K. Chehdi, and A. Nasri. Multi-component image segmentation using hybrid dynamic genetic algorithm and fuzzy c-means. *IET image processing*, 3(2):52–62, 2009.

[AFO+08] R. B. Arajo, G. H. T. Ferreira, G. H. Orair, W. Meira, R. A. C. Ferreira, D. O. G. Neto, and M. J. Zaki. The partricluster algorithm for gene expression analysis. *International Journal of Parallel Programming*, 36(2):226–249, 2008.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of SIGMOD'98*, pages 94–105, Seattle, 1998.

[AK03] Hanan Ayad and Mohamed Kamel. Finding natural clusters using multi-clusterer combiner based on shared nearest neighbors. In *Proceedings of the 4th international conference on Multiple classifier systems*, pages 166–175, Berlin, Heidelberg, 2003. Springer-Verlag.

[AK07] M. Acharyya and M.K. Kundu. Image segmentation using wavelet packet frames and neuro-fuzzy tools. *IEEE Transactions of Computational Cognition*, 5(4):27–41, 2007.

[ALKK07] Lamine M. Aouad, Nhien-An Le-Khac, and Tahar M. Kechadi. Lightweight clustering technique for distributed data mining applications. In *Proceedings of the 7th industrial conference on Advances in data mining: theoretical aspects and applications*, ICDM'07, pages 120–134, Berlin, Heidelberg, 2007. Springer-Verlag.

[AN09] M.M. Awad and A. Nasri. Satellite image segmentation using self- organizing maps and fuzzy c-means. In *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 2009*, pages 398 –402, 2009.

[ANS06] S. Asharaf, M. Narasimha, and S.K. Shevade. Rough set based incremental clustering of interval data. *Pattern Recognition Letters*, 27:515–519, 2006.

[Ast70] M.M. Astrahan. Speech analysis by clustering, or the hyper-phoneme method. Stanford A. I. Project Memo, 1970.

[AW10] Ed. Charu Aggarwal and Haixun Wang, editors. *Managing and Mining Graph Data.* Springer, 2010.

[AWY+99] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, SIGMOD '99, pages 61–72, New York, NY, USA, 1999. ACM.

[AY00] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. *SIGMOD Record*, 29:70–81, 2000.

[AZM06] F. Ameri, M. J. V. Zoej, and M. Mokhtarzade. Satellite image segmentation based on fuzzy c-means clustering. In *Proceedings of Map Asia*, 2006.

[B+03] F. G. Berriz et al. Characterizing gene sets with funcassociate. *Bioinformatics*, 19:2502–2504, 2003.

[BB07] B. Borah and D.K. Bhattacharyya. A clustering technique using density difference. In *Proceedings of International Conference on Signal Processing, Communications and Networking*, pages 585–588, 2007.

[BB08] B. Borah and D.K. Bhattacharyya. DDSC: A density differentiated spatial clustering technique. *Journal of Computers*, 3(2):72–79, 2008.

[BBD04] B. Borah, D. K. Bhattacharyya, and R. K. Das. A parallelization of density based clustering technique on distributed memory multicomputer. In *Proceedings of ADCOM*, pages 536–541, Ahmedabad, 2004.

[BD08] A. Bhattacharya and R. De. Divisive correlation clustering algorithm (dcca) for grouping of genes: detecting varying patterns in expression profiles. *Bioinformatics*, 24(11):1359–1366, 2008.

[BdCKY02] Amir Ben-dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. In *Proceedings of the 6th International Conference on Computational Biology (RECOMB02)*, pages 49–57, 2002.

[BDSY99] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3-4):281–297, 1999.

[Bez81a] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum, New York, USA, 1981.

[Bez81b] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

[BG03] M.J. Beal and Z. Ghahramani. The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures. In *Proceedings of the 7th Valencia International Meeting on Bayesian Statistics*, volume 63(4), pages 453–464, Spain, 2003.

[BGM+06] Sanghamitra Bandyopadhyay, Chris Giannella, Ujjwal Maulik, Hillol Kargupta, Kun Liu, and Souptik Datta. Clustering distributed data streams in peer-to-peer environments. *Information Sciences*, 176(14):1952 – 1985, 2006.

[BH67] G.H. Ball and D.J. Hall. A clustering technique for summarizing multivariate data. *Behavioural Science*, 12:153–155, 1967.

[Bic01] D.R. Bickel. Robust cluster analysis of dna microarray data: An application of nonparametric correlation dissimilarity. In *Proceedings of the Joint Statistical Meetings of the American Statistical Association (Biometrics Section)*, 2001.

[BKNS00] Markus Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jrg Sander. LOF: Identifying density-based local outliers. In *Proceedings of SIGMOD*, pages 93–104, 2000.

[BL94] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley & Sons, 1994.

[BM93] G. Babu and M. Murty. A near-optimal initial seed value selection in k-means algorithm using a genetic algorithm. *Pattern Recognition Letters*, 14(10):763–769, 1993.

[BMM07a] S. Bandyopadhyay, U. Maulik, and A. Mukhopadhyay. Multiobjective genetic clustering for pixel classification in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 45(2):1506–1511, 2007.

[BMM07b] S. Bandyopadhyay, A. Mukhopadhyay, and U. Maulik. An improved algorithm for clustering gene expression data. *Bioinformatics*, 23(21):2859–2865, 2007.

[BP01] S. Bandyopadhyay and S.K. Pal. Pixel classification using variable string genetic algorithms with chromosomal differentiation. *IEEE Transactions on Geoscience and Remote Sensing*, 39(2):303–308, 2001.

[BPC02] A. Bellaachia, D. Portnoy, and A. G. Chen, Y.and Elkahloun. E-cast: A data mining algorithm for gene expression data. In *Proceedings of the BIOKDD02: Workshop on Data Mining in Bioinformatics (with SIGKDD02 Conference)*, page 49, 2002.

[CC00] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press, 2000.

[CCFM97] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 626–635, New York, NY, USA, 1997. ACM.

[CCW+98] R. J. Cho, M. Campbell, E. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. Wolfsberg, A. Gabrielian, D. Landsman, and D. Lockart. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2(1):65–73, 1998.

[CFZ99] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 84–93, New York, NY, USA, 1999. ACM.

[CH07] D. Cook and L. Holder. *Mining Graph Data.* John Wiley & Sons Inc, 2007.

[Che03] Yiu-Ming Cheung. k*-means: A new generalized k-means clustering algorithm. *Pattern Recognition Letters*, 24(15):2883 - 2893, 2003.

[CHNW96] D.W. Cheung, J. Han, V.T. Ng, and Y. Wong. Maintenance of discovered association rules in large databases: An incremental technique.

In *Proceedings of* 12^{th} *International Conference on Data Engineering*, pages 106–114, New Orleans, USA, 1996.

[CHO02] Chien-Yu Chen, Shien-Ching Hwang, and Yen-Jen Oyang. An incremental hierarchical data clustering algorithm based on gravity theory. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '02, pages 237–250, London, UK, 2002. Springer-Verlag.

[CJ02] Jae-Woo Chang and Du-Seok Jin. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 503–507, New York, NY, USA, 2002. ACM.

[CJM04] S. Chung, J. Jun, and D. McLeod. Mining gene expression datasets using density based clustering. Technical Report IMSC-04-002, USC/IMSC, University of Southern California, 2004.

[CS96] Peter Cheeseman and John Stutz. Advances in knowledge discovery and data mining, 1996.

[DBK09a] R. Das, D.K. Bhattacharyya, and J.K. Kalita. Clustering gene expression data using a regulation based density clustering. *International Journal of Recent Trends in Engineering*, 2(1-6):76–78, 2009.

[DBK09b] R. Das, D.K. Bhattacharyya, and J.K. Kalita. An incremental clustering of gene expression data. In *Proceedings of NABIC*, pages 742–747, Coimbatore, India, 2009.

[DBK10] R. Das, D.K. Bhattacharyya, and J.K. Kalita. Clustering gene expression data using an effective dissimilarity measure. International Journal of Computational BioScience (Special Issue), 2010.

[DC97] J. Dopazo and JM. Carazo. Phylogenetic reconstruction using an unsupervised neural network that adopts the topology of a phylogenetic tree. *Journal of Molecular Evolution*, 44:226–233, 1997.

[DGK06] S. Datta, C. Giannella, and H. Kargupta. K-Means Clustering over a Large, Dynamic Network. In *Proceedings of 2006 SIAM Conference on Data Mining*, Bethesda, MD, April 2006.

[DGK09] Souptik Datta, Chris Giannella, and Hillol Kargupta. Approximate distributed k-means clustering over a peer-to-peer network. *IEEE Transactions on Knowledge and Data Engineering*, 21:1372–1388, 2009.

[DI97] J.L. DeRisi and P.O. Iyer, V.R.and Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686, 1997.

[DK03] D. Dembl and P. Kastner. Fuzzy C-means method for clustering microarray data. *Bioinformatics*, 19(8):973–980, 2003.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 (1):1–38, 1977.

[DM99] I. S. Dhilon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD 99)*, San Diego, CA, USA, 1999.

[DXLG06] L. Duan, D. Xiong, J. Lee, and F. Guo. A local density based spatial clustering algorithm with noise. In *IEEE International Conference on Systems, Man, and Cybernetics*, Taipei, Taiwan, 2006.

[ECL00] V. Estivill-Castro and I. Lee. Amoeba: Hierarchical clustering based on spatial proximity using delaunay diagram. In *Proceedings of the 9th International Symposium on Spatial Data Handling*, 2000.

[EKS+98] M. Ester, H. P. Kriegel, J. Sander, M. Wimmer, and X. Xu. An incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24th VLDB Conference*, New York, USA, 1998.

[EKSX96] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of International Conference on Knowledge Discovery in Databases and Data Mining (KDD-96)*, pages 226–231, Portland, Oregon, 1996.

[ESBB98] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. In *Proceedings of National Academy of Sciences*, volume 95, pages 14863–14868, 1998.

[ESK03] L. Ertoz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy high dimensional data. In *SIAM International Conference on Data Mining*, 2003.

[FAAM97] R. Feldman, Y. Aumann, A. Amir, and H. Mannila. Efficient algorithms for discovering frequent sets in incremental databases. In *Proceedings of ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 59–66, Tucson, AZ, 1997.

[FJ02] Ana L.N. Fred and Anil K. Jain. Data clustering using evidence accumulation. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, volume 4 of *ICPR '02*, pages 276–280, Washington, DC, USA, 2002. IEEE Computer Society.

[FLPT00] D. Foti, D. Lipari, C. Pizzuti, and D. Talia. Scalable parallel clustering for data mining on multicomputers. In *Lecture Notes in Computer Science*, pages 390–398. Springer Verlag, 2000.

[FM04] Jerome H. Friedman and Jacqueline J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society*, 66:815–849, 2004.

[GGN+99] Sanjay Goil, Sanjay Goil, Harsha Nagesh, Harsha Nagesh, Alok Choudhary, and Alok Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. Technical report, Northwestern University, 2145 Sheridan Road, Evanston IL 60208, 1999.

[GLD00] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences of the United States of America*, 97(22):12079–12084, 2000.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, New York, 1989.

[GR02] F. Gibbons and F. Roth. Judging the quality of gene expression based clustering methods using gene annotation. *Genome Research*, 12:1574–1581, 2002.

[GRG+98] Venkatesh Ganti, Raghu Ramakrishnan, Johannes Gehrke, Allison Powell, and James French. Clustering large datasets in arbitrary metric spaces. In *IN PROCeedings OF THE 15TH INTernational CONFerence ON DATA ENGineering*, pages 502–511, 1998.

[GRS98] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *SIGMOD Record*, volume 27(2), pages 73–84, 1998.

[GRS99] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. In *In Proceedings of the 15th International Conference on Data Engineering*, pages 512–521, 1999.

[GyFSlXr09] D. Gen-yuan, M. Fang, T. Sheng-li, and G. Xi-rong. Remote sensing image sequence segmentation based on the modified fuzzy c-means. *Journal of Software*, 5(1), 2009.

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2):147160, 1950.

[HC04] C. Hsu and M. Chen. Subspace clustering of high dimensional spatial data with noises. In *Proceedings of PAKDD 04*, pages 31–40, 2004.

[HCFdC09] Eduardo Raul Hruschka, Ricardo J. G. B. Campello, Alex A. Freitas, and Andr C. Ponce Leon F. de Carvalho. A survey of evolutionary algorithms for clustering. *IEEE TRANSACTIONS ON SYSTEMS, MAN,*

AND CYBERNETICSPART C: APPLICATIONS AND REVIEWS, 39(2), 2009.

[HDRT04] F. V. D. Heijden, R. Duin, D. Ridder, and D. M. J. Tax. *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB.* John Wiley and Sons, 2004.

[HHK98] Alexander Hinneburg, Er Hinneburg, and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 58–65. AAAI Press, 1998.

[HK06] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, San Fransisco, USA, 2006.

[HK07] J. Handl and J. Knowles. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computing*, 11(1):56–76, 2007.

[HKK05] J. Handl, J. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21:3201–3212, 2005.

[HKY99] L.J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identication and analysis of co-expressed genes. *Genome Research*, 9(11):11061115, 1999.

[HMS04] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* Prentice-Hall of India, New Delhi, 2004.

[HOB99] L.O. HALL, B. OZYURT, and J.C. BEZDEK. Clustering with a genetically optimized approach. *IEEE Transactions on Evolutionary Computation*, 3(2):103–112, 1999.

[HSL+99] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, and R. Shamir. An algorithm for clustering cDNAs for gene expression analysis using short oligonucleotide fingerprints. In *Proceedings of 3rd International Symposium on Computational Molecular Biology (RECOMB 99)*, pages 188–197. ACM Press, 1999.

[Hua98] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[HVD01] J. Herrero, A. Valencia, and J. Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17:126136, 2001.

[IER+99] V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J. Lee, J.M. Trent, L.M. Staudt, J.J. Hudson, M.S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P.O. Brown. The transcriptional program in the response of the human fibroblasts to serum. *Science*, 283:83–87, 1999.

[JGA06] Ruoming Jin, Anjan Goswami, and Gagan Agrawal. Fast and exact out-of-core and distributed k-means clustering. *Knowledge and Information Systems*, 10:17–40, 2006.

[JK99] Erik L. Johnson and Hillol Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data, 1999.

[JKN98] T. Johnson, I. Kwok, and R.T. Ng. Fast computation of 2-dimensional depth contours. In *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 224–228, 1998.

[JKP03] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. Towards effective and efficient distributed clustering. In *In Workshop on Clustering Large Data Sets (ICDM)*, pages 49–58, Melbourne, Florida, 2003.

[JMF99] A.K. Jain, M.N. Murty, and P.J. Flyn. Data clustering: A review. *ACM Computing Surveys*, 31(3), 1999.

[JP73] R.A. Jarvis and E.A. Patrick. Clustering using a similarity measure based on shared nearest neighbors. *IEEE Transactions on Computers*, 11, 1973.

[JPZ03] D. Jiang, J. Pei, and A. Zhang. DHC: a density-based hierarchical clustering method for time series gene expression data. In *Proceedings of BIBE2003: 3rd IEEE International Symposium on Bioinformatics and Bioengineering*, page 393, Bethesda, Maryland, USA, 2003.

[JTZ04] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, 2004.

[KGX+06] H. S. Kim, S. Gao, Y. Xia, G.B. Kim, and H. Y. Bae. Dgcl: An efficient density and grid based clustering algorithm for large spatial database. In *Advances in Web-Age Information Management (WAIM 2006)*, pages 362–371, 2006.

[KHK99] G. Karypis, J. Han, and V. Kumar. CHAMELEON: A hierarchical clustering algorithm using dynamic modelling. *IEEE Computer*, 32(8):68–75, 1999.

[KM99] K. Krishna and M. Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 29:433–439, 1999.

[KN98] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 392–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[KNT00] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *The Very Large Data Bases Journal*, 8(3):237–253, 2000.

[KNZ01] Edwin M. Knorr, Raymond T. Ng, and Ruben H. Zamar. Robust space transformations for distance-based operations. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135, New York, NY, USA, 2001. ACM.

[Koh95] T. Kohonen. *Self-organizing maps.* Springer-Verlag, Heidelberg, Germany, 1995.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis* John Wiley & Sons, 1990.

[KW02] S. A. Krawetz and D. D. Womble. *Introduction to Bioinformatics: A Theoretical and Practical Approach.* Humana Press, Totowa, NJ, USA, 2002.

[LEC00] I. Lee and V. Estivil-Castro. Autoclust: Automatic clustering via boundary extraction for mining massive point data sets. In *Proceedings of the 5th International Conference on Geocomputation*, 2000.

[LKAK07] Nhien-An Le-Khac, Lamine M. Aouad, and M-Tahar Kechadi. A new approach for distributed density based clustering on grid platform. In *Proceedings of the 24th British national conference on Databases*, BNCOD'07, pages 247–258, Berlin, Heidelberg, 2007. Springer-Verlag.

[LLF+04a] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S.J. Brown. FGKA: A fast genetic k-means algorithm. In *Proc. ACM Symposium on Applied Computing*, 2004.

[LLF+04b] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S.J. Brown. Incremental genetic k-means algorithm and its application in gene expression data analysis. *BMC Bioinformatics*, 5(172), 2004.

[LT09] Ao Li and David Tuck. An effective tri-clustering algorithm combining expression data with gene regulation information. *Gene Regulation and Systems Biology*, 3:49–64, 2009.

[LWN+09] G. Li, Z. Wang, Q. Ni, X. Wang, B. Qiang, and H. Qing-juan. Application of a new similarity measure in clustering gene expression data. Downloaded from: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05162382, 2009.

[LZW07] P. Liu, D. Zhou, and N. Wu. Vdbscan: Varied density based spatial clustering of applications with noise. In *IEEE*, 2007.

[MB03a] U. Maulik and S. Bandyopadhyay. Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. *IEEE. Transactions on Geoscience and Remote Sensing*, 41(5):1075–1081, 2003.

[MB03b] U. Maulik and S. Bandyopadhyay. Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. *IEEE Transactions on Geoscience and Remote Sensing*, 41:1075–1081, 2003.

[McQ67] J.B. McQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[MMB09] U. Maulik, A. Mukhopadhyay, and S. Bandyopadhyay. Combining pareto-optimal clusters using supervised learning for identifying co-expressed genes. *BMC Bioinformatics*, 10(27), 2009.

[MMP02] P. Mitra, C. A. Murthy, and S. K. Pal. Density-based multiscale data condensation. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 24(6), June 2002.

[MO04] S.C. Madeira and A.L. Oliveria. Biclustering algorithms for biological data analysis: A survey. *IEEE*, 1(1), 2004.

[NGC00] H. S. Nagesh, S. Goil, and A. N. Choudhary. A scalable parallel subspace clustering algorithm for massive data sets. In *Proceedings of International Conference on Parallel Processing*, page 477, 2000.

[NH02] R. Ng and J. Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002.

[PC02] P. Pal and B. Chanda. A symmetry based clustering technique for multi-spectral satellite imagery, 2002.

[Per01] W. Perrizo. Peano count tree technology. Technical report, NDSU-CSOR-TR-01-1, North Dakota State University, Fargo, North Dakota, United States, 2001.

[PGS00] S.K. Pal, A. Ghosh, and B. U. Shankar. Segmentation with remotely sensed images with fuzzy thresholding and quantitative evaluation. *International Journal of Remote Sensing*, 21(11):2269–2300, 2000.

[PJAM02] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, SIGMOD '02, pages 418–427, New York, NY, USA, 2002. ACM.

[RB05] S. Roy and D. K. Bhattacharyya. An approach to find embedded clusters using density based techniques. In *Proceedings of the ICDCIT*, pages 523–535, 2005.

[RJJK10] A. Ram, S. Jalal, A. S. Jalal, and M. Kumar. DVBSCAN: A density variation based spatial clustering of applications with noise. *International Journal of Computer Applications*, 3(6), 2010.

[RR96] I. Ruts and P. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.

[RRAR06] R. Ruiz, J.C. Riquelme, and J.S. Aguilar-Ruiz. Incremental wrapper-based gene selection from microarray data for cancer classification. *Pattern Recognition*, 39:23832392, 2006.

[RRS00] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 427–438, 2000.

[SAP06] R. Syamala, T. Abidin, and W. Perrizo. Clustering microarray data based on density and shared nearest neighbor measure. In *Computers and Their Applications*, pages 360–365, 2006.

[SB10] S. Sarmah and D. K. Bhattacharyya. Disclus: A distributed clustering technique over high resolution satellite data. In *Proceedings of ICDCN 2010*, 2010.

[SC+98] G. Sheikholeslami, S. Chatterjee, et al. Wavecluster: A multi- resolution clustering approach for very large spatial database. In *Proceedings of SIGMOD'98*, Seattle, 1998.

[SDB08] S. Sarmah, R. Das, and D. K. Bhattacharyya. A distributed algorithm for intrinsic cluster detection over large spatial data. *International Journal of Computer Science*, 3(4):246–256, 2008.

[SEKX98] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[SG03] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2003.

[SMKS03] R. Sharan, A. Maron-Katz, and R. Shamir. Click and expander: a system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, 2003.

[SS00] R. Sharan and R. Shamir. Click: A clustering algorithm with applications to gene expression analysis. In *Proceedings of 8th International Conference on Intelligent Systems for Molecular Biology*, pages 307–316. AAAI Press, 2000.

[Ste06] D. Stekel. *Microarray Bioinformatics.* Cambridge University Press, Cambridge, UK., 2006.

[SZCS03] G. Shu, B. Zeng, Y.P. Chen, and O.H. Smith. Performance assessment of kernel density clustering for gene expression prole data. *Comparative and Functional Genomics*, 4:287299, 2003.

[TBK09] Luis Tari, Chitta Baral, and Seungchan Kim. Fuzzy c-means clustering with prior biological knowledge. *Journal of Biomedical Informatics*, 42(1):74–81, 2009.

[TH09] J.H. Travis and Y. Huang. Clustering of gene expression data based on shape similarity. *EURASIP Journal on Bioinformatics and Systems Biology,*, 2009(195712), 2009.

[THC+99] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, and G. Church. Systematic determination of genetic network architecture. *Nature Genet*, 22:281285, 1999.

[THHK02] S. Tomida, T. Hanai, H. Honda, and T. Kobayashi. Analysis of expression profile using fuzzy adaptive resonance theory. *Bioinformatics*, 18(8):1073–83, 2002.

[TMEDF08] V. Torra, S. Miyamoto, Y. Endo, and J. Domingo-Ferrer. On intuitionistic fuzzy clustering for its application to privacy. In *Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence). IEEE International Conference on*, pages 1042 –1048, 2008.

[TSK09] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Pearson Education, New York, 2009.

[TSS02] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics (Oxford, England)*, 18 Suppl 1:S136S144, 2002.

[TZRZ01] Chun Tang, Li Zhang, Murali Ramanathan, and Aidong Zhang. Interrelated two-way clustering: An unsupervised approach for gene expression data analysis. In *Proceedings of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering*, pages 41–48, Washington, DC, USA, 2001. IEEE Computer Society.

[VB09] P. Viswanath and V. Suresh Babu. Rough-dbscan: A fast hybrid density based clustering method for large data sets. *Pattern Recognition Letters*, 30:14771488, 2009.

[VTP10] N. Karthikeyani Visalakshi, K. Thangavel, and R. Parvathi. An intuitionistic fuzzy approach to distributed fuzzy clustering. *International Journal of Computer Theory and Engineering*, 2(2):1793–8201, 2010.

[WYM97] W. Wang, J. Yang, and R. R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of VLDB 97*, pages 186 – 195, Athens, Greece, 1997.

[XEKS98] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. A nonparametric clustering algorithm for knowledge discovery in large spatial data sets. In *Proceedings of IEEE International Conference on Data Engineering*. IEEE Computer Society Press, 1998.

[XJK99] Xiaowei Xu, Jochen Jäger, and Hans-Peter Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Mining and Knowledge Discovery*, 3(3):263–290, 1999.

[YAL+06] H.-S. Yoon, S.-Y. Ahn, S.-H. Lee, S.-B. Cho, and J. H. Kim. Heterogeneous clustering ensemble method for combining different cluster results. In *Proceedings of BioDM, Lecture Notes in Computer Science*, volume 3916, page 8292, 2006.

[Yam98a] T. Yamazaki. A robust clustering technique for multi-spectral satellite images. In *Proceedings of the International Symposium on Noise Reduction for Imaging and Communication Systems (ISNIC)*, 1998.

[Yam98b] T. Yamazaki. A robust clustering technique for multi-spectral satellite images, 1998.

[YcMFJd03] Z. Yan-chang, S. Mei, X. Fan, and S. Jun-de. Clustering datasets containing clusters of various densities. *Journal of Beijing University of Posts and Telecommunications*, 26(2):42–47, 2003

[YHCY10] T. Yun, T. Hwang, K. Cha, and GS. Yi. CLIC: clustering analysis of large microarray datasets with individual dimension-based clustering. *Nucleic Acids Research*, 38:W246–W253, 2010.

[YJ01] Z. Yanchang and S. Junde. Gdilc: A grid-based density-isoline clustering algorithm. *IEEE*, 2001.

[YWWY02] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Proceedings of 18th International Conference on Data Engineering*, pages 517–528, 2002.

[YWWY03] Jiong Yang, Haixun Wang, Wei Wang, and Philip Yu. Enhanced biclustering on expression data. In *Proceedings of the 3rd IEEE Symposium on BioInformatics and BioEngineering*, BIBE '03, pages 321–327, Washington, DC, USA, 2003. IEEE Computer Society.

[ZCK99] Bo Zhou, David W. Cheung, and Ben Kao. A fast algorithm for density-based clustering in large database. In *Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 338–349. Springer-Verlag., 1999.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *Proceedings of ACM-SIGMOD International Conference Management of Data*, pages 103–114, Montreal, Canada, 1996.

[ZZ05] L. Zhao and M.J. Zaki. tricluster: An effective algorithm for mining coherent clusters in 3d microarray data. In *In Proc. of the 2005 ACM SIGMOD international conference on Management of data*, pages 694–705. ACM Press, 2005.

List of Publications

1. Communicated Papers:

   (a) Sauravjyoti Sarmah, Rosy Das and D.K. Bhattacharyya, *An Effective Density-based Hierarchical Clustering Technique to identify Coherent Patterns from Gene Expression Data*, Communicated to the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2011.

   (b) Sauravjyoti Sarmah and D.K. Bhattacharyya, *A Grid-density based Technique for Clustering Satellite Image*, Communicated to the Journal of Pattern Recognition Letters, 2010.

2. Journal Papers:

   (a) Sauravjyoti Sarmah and D.K. Bhattacharyya, *An Effective Technique for Clustering Incremental Gene Expression data*, International Journal of Computer Science Issues (IJCSI), Vol. 7, Issue 3, No 3, pp. 31-41, May 2010.

   (b) Sauravjyoti Sarmah, R. Das and D.K. Bhattacharyya, *A Distributed Algorithm for Intrinsic Cluster Detection over Large Spatial Data*, International Journal of Electrical and Computer Engineering, vol 3(4), Fall 2008, pp. 246-256.

3. Conference Papers:

   (a) Sauravjyoti Sarmah and D.K. Bhattacharyya, *DisClus: A Distributed Clustering Technique over High Resolution Satellite Data*, 11th International Conference on Distributed Computing and Networking, Published

as a book chapter in the Distributed Computing and Networking, LNCS 5935, ISSN 0302-9743, pp 353-364, 2010, Springer-Verlag, Kolkata, India.

(b) Rory Lewis, J.K. Kalita, Sauravjyoti Sarmah and D.K. Bhattacharyya, *Music Industry Scalar Analysis Using Unsupervised Fourier Feature Selection*, 17th International Conference on Intelligent Information Systems, pp. 485-494, IIS 2009, Poland.

(c) Sauravjyoti Sarmah, R. Das and D.K. Bhattacharyya, *DGDCT: A Distributed Grid-Density based Algorithm for Intrinsic Cluster Detection over Massive Spatial Data*, International Conference on Distributed Computing and Networking (ICDCN 2008), Published as a book chapter in the Distributed Computing and Networking, LNCS ISSN 0302-9743, pp 239-250, Kolkata, India.

(d) Sauravjyoti Sarmah and D.K. Bhattacharyya, *Grid-based Clustering Technique for Satellite Image Application*, in the Proceedings of INCA2008, Ahmedabad, India.

(e) Sauravjyoti Sarmah, R. Das and D.K. Bhattacharyya, *Intrinsic Cluster Detection Using Adaptive Grids*, The Fifteenth International Conference On Advanced Computing & Communications (ADCOM 2007), pp. 371-376, 2007, Guwahati, India.

(f) Sauravjyoti Sarmah and D.K. Bhattacharyya, *A Faster Clustering Technique for Intrinsic Cluster Detection over Spatial Datasets*, National Conference on Trends in Advanced Computing (NCTAC 2007), pp 147-154, Tezpur, India.

(g) Sauravjyoti Sarmah, R. Das and D.K. Bhattacharyya, *A Distributed Clustering Technique for Intrinsic Cluster Detection*, The Fourteenth International Conference On Advanced Computing & Communications (ADCOM 2006), pp 117-122, Mangalore, India.