AN EFFICIENT EVOLUTIONARY CLUSTERING AND PREDICTION MODEL FOR GENE EXPRESSION TIME SERIES DATA

by

Atakan ERDEM

B.S., Computer Engineering, Bilkent University, 1995

M.S., Computer Engineering, METU, 1998

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering
Boğaziçi University
2014

AN EFFICIENT EVOLUTIONARY CLUSTERING AND PREDICTION MODEL FOR GENE EXPRESSION TIME SERIES DATA

APPROVED BY:

Prof. Taflan Gündem (Thesis Supervisor)	.....
Prof. Fikret Gürgen	.....
Prof. Osman Uğur Sezerman	.....
Assoc. Prof. Yücel Saygın	.....
Assist. Prof. İsmail Arı	.....

DATE OF APPROVAL: 05.06.2014

ACKNOWLEDGEMENTS

There are numerous people I would like to thank due to their support and encouragement in completion of this thesis. First of all, all my gratitude goes to my supervisor Prof. Taflan Gündem. He has truly inspired me during my research, provided helpful hints and always supported me through the good and the bad times. So, it has been a pleasure to work with him. I also would like to thank Prof. Osman Uğur Sezerman for always being helpful and encouraging to me. My thanks also go to my thesis jury members Prof. Fikret Gürgen, Assoc. Prof. Yücel Saygın and Assist. Prof. İsmail Arı.for spending their rare available time on examining my thesis.

I am indebted to my institution TÜBİTAK-BİLGEM for motivating its employees to pursue an academic career and funding research expenditures. Special thanks also go to my friends and colleagues for their altruistic supports to the visual arrangement of the thesis report.

Last but not least I would like to mention my parents, for their unconditional support. They have always encouraged me when I took important decisions and always been with me when I needed them. Finally, my deepest thanks go to my wife, Filiz and my lovely daughters Bilge and Ceyda for their love and their enduring support. Without their patience and encouragement, this thesis would never have been written.

ABSTRACT

AN EFFICIENT EVOLUTIONARY CLUSTERING AND PREDICTION MODEL FOR GENE EXPRESSION TIME SERIES DATA

Because of using manual methods in some parts of gene expression experiments, reliability of the data is low. If this data is directly utilized as input to a data mining algorithm or a model for evaluating gene expression data, then the adverse affects to the desired results will be inevitable. In order to eliminate aforementioned adverse affects and reduce the fuzziness, we represent the data with sample data sets that are generated by using uncertain data management techniques. Sample data approach not only reduces the percentage of fuzziness, but also it causes the output generation time to be increased due to an increase in the amount of processed data, which is directly proportional to the cardinality of the sample data set. In the first part of the study, we introduce an uncertain data clustering algorithm, named *M-FDBSCAN*, for enabling one to cluster uncertain data rapidly, which runs on multi-core systems in a concurrent fashion. We show that by using the proposed method, the algorithm yields considerable performance improvement on single core systems, as well. In the second part of the study, *M-FDBSCAN* algorithm is converted into an evolutionary clustering algorithm, named *E-MFDBSCAN*, by which time series data can be processed rapidly and efficiently. This new algorithm enables to generate global clusters. In the last part of the study, using time-based evolutionary patterns of global clusters a prediction model is constructed. The proposed prediction model enables us to predict the patterns and the similarities of a global cluster that will be generated at the next time point.

ÖZET

GEN İFADESİ ZAMAN SERİSİ VERİLERİ İÇİN ETKİN BİR EVRİMSEL KÜMELEME VE ÖNGÖRÜ MODELİ

Gen ifadesi deneylerinin bir aşamasında veriler manuel yöntemlerle elde edildiği için verilerin güvenilirliği düşüktür. Bu verilerin bir veri madenciliği algoritmasına ya da modele direkt girdi olması durumunda varılmak istenen sonuçların güvenilirliğinin olumsuz yönde etkilemesi kaçınılmazdır. Çalışmamızda, elde edilen verilerin belirsizliğini azaltmak için her verinin, örnek veri üretme teknikleriyle elde edilen veri kümeleriyle temsil edilmesini sağladık. Örnek veri yaklaşımı verilerin belirsizlik yüzdesini azaltırken işlem yapılan veri setinin örnek veri kümesi eleman sayısı oranında artmasına, dolayısıyla da ilgili veri işleme algoritmalarının sonuç üretme zamanının artmasına neden olmaktadır. Çalışmamızın ilk kısmında belirsiz verilerin hızlı bir biçimde kümelenebilmesi için çok çekirdekli sistemler üzerinde eş zamanlı çalışabilen *M-FDBSCAN* adını verdiğimiz bir "belirsiz veri kümeleme" algoritması geliştirdik. Algoritmada önerilen yöntemle yalnızca çok çekirdekli sistemlerde değil tek çekirdekli sistemlerde de veri işleme hızında büyük artışlar sağlandığı gösterdik. Çalışmamızın ikinci kısmında *M-FDBSCAN* algoritmasını, zaman serisi verilerinin hızlı ve etkin bir biçimde işlenebildiği, *E-MFDBSCAN* adı verilen bir "evrimsel kümeleme" algoritmasına dönüştürdük. Bu yeni algoritma global kümelerin oluşturulmasını sağlamaktadır. Çalışmamızın son aşamasında oluşturulan global kümelerin zaman bazlı evrimsel desenlerini kullanarak bir öngörü modeli geliştirdik. Bu öngörü modeliyle bir sonraki zaman noktasına ait bir global kümenin benzerlik ve desen bilgilerinin kestiriminin yapılabilmesini sağladık.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xii
LIST OF ACRONYMS/ABBREVIATIONS . . . xiii
1. INTRODUCTION . . . 1
2. BACKGROUND . . . 5
2.1. Proteins . . . 5
2.1.1. Protein Function . . . 6
2.1.2. Protein Sequences . . . 7
2.2. Gene Expression Data . . . 8
3. RELATED APPROACHES AND ALGORITHMS . . . 12
3.1. Uncertain Data Management and Mining . . . 12
3.2. Density-Based Uncertain Data Clustering . . . 14
3.3. FDBSCAN . . . 17
3.3.1. Definitions . . . 17
3.3.2. The Algorithm . . . 18
3.4. Clustering of Gene Expression Time-Series Data . . . 20
3.4.1. Time-series . . . 21
3.4.2. Gene expression time-series . . . 22
3.4.3. Similarity of gene expression time-series . . . 22
3.4.3.1. Similarity requirements for co-expression . . . 23
3.4.4. Clustering of gene expression time-series . . . 24
4. THESIS STUDY . . . 27
4.1. M-FDBSCAN . . . 27
4.1.1. Computational Aspects . . . 28
4.1.2. The Algorithm . . . 31

4.1.2.1. Dataset splitting 31
4.1.2.2. Concurrent clustering 32
4.1.2.3. Merging sub data regions 33
4.1.3. Pseudocodes 37
4.1.4. Experimental study 41
4.2. An Evolutionary Clustering and Prediction Model for Gene Expression Data 46
4.2.1. E-MFDBSCAN 47
4.2.1.1. The Algorithm 53
4.2.2. The Prediction Model 55
4.2.2.1. BestMatch 57
4.2.3. Results and discussion 57
5. CONCLUSION 61
REFERENCES 63

LIST OF FIGURES

Figure 2.1. A possible hierarchical organization of the categories of protein function. 7

Figure 2.2. The central dogma of molecular biology: conversion of gene to protein via mRNA. 8

Figure 2.3. An illustration of the microarray experimental procedure. 10

Figure 3.1. Density-Based profile with lower density threshold. 14

Figure 3.2. Density-Based profile with higher density threshold. 15

Figure 3.3. Minimum Bounding Rectangles for X and Y fuzzy objects where s=3. 19

Figure 3.4. A 4×4 sample matrix. 20

Figure 4.1. A 4×4 sample matrix illustration without splitting dataset. . . . 29

Figure 4.2. A 4×4 sample matrix illustration with splitting dataset into 4 sub datasets. 29

Figure 4.3. Merging sub-data regions algorithm. 34

Figure 4.4. Case 1: f1 and f2 belong to different clusters. 35

Figure 4.5. Case 2: f1 is outliers and f2 belongs to a cluster. 35

Figure 4.6. Case 3: f1 belongs a cluster and f2 is outliers. 36

Figure 4.7. Case 4: f1 and f2 are both outliers. 36

Figure 4.8. An illustration of dataset splitting. 37

Figure 4.9. *M-FDBSCAN* algorithm. 38

Figure 4.10. *Split Dataset* procedure. 39

Figure 4.11. *Merge Dataset* procedure. 40

Figure 4.12. For 50,000 *fdo* records, the execution times of *FDBSCAN*, *M-FDBSCAN* (single-core), and *M-FDBSCAN* (multi-core). 44

Figure 4.13. For 8 splits, the merging times needed for *M-FDBSCAN* for single-core and *M-FDBSCAN* for multi-core systems. 45

Figure 4.14. A *t-sample matrix* with $m = 5$ time points and $s = 3$ sample data points. 49

Figure 4.15. Procedure to compute *Core Object Probability*. 50

Figure 4.16. Procedure to compute *Distance Distribution Probability*. 51

Figure 4.17. An illustration of gene expression time series database. 52

Figure 4.18. Gene expression time series database with *splitting lines*. 53

Figure 4.19. *E-MFDBSCAN* procedure. 54

Figure 4.20. Illustration of time-based evolutionary change of sets of clusters. . 56

Figure 4.21. Similarities vs. Time graph for budding yeast dataset. 58

Figure 4.22. Similarities vs. Time graph for breast cancer dataset. 59

LIST OF TABLES

Table 4.1. Observation results. 42

LIST OF SYMBOLS

c	Number of cores/splits
$d_{\mathbf{max}}$ (x,y)	Maximum distance between two fuzzy data objects, x and y
$d_{\mathbf{min}}$(x,y)	Minimum distance between two fuzzy data objects, x and y
N_{ε} (x)	Neighborhood function returns the neighbors of x in ε-neighborhood
s	Cardinality of a sample data set
ε	Maximum distance between two members of a cluster
μ	Minimum number of members in a cluster

LIST OF ACRONYMS/ABBREVIATIONS

AR	Autoregressive
cDNA	Complementary DNA
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DNA	Deoxyribonucleic acid
E-MFDBSCAN	Evolutionary Multi-core Fuzzy Density-Based Spatial Clustering of Applications with Noise
FDBSCAN	Fuzzy Density-Based Spatial Clustering of Applications with Noise
fdo	fuzzy data object
MBR(o)	Minimum Bounding Rectangle of a fuzzy data object o
mRNA	Messenger RNA
M-FDBSCAN	Multi-core Fuzzy Density-Based Spatial Clustering of Applications with Noise
Omics	A measurement of the totality of all molecules, interactions and functions within given levels of cellular processes
RNA	Ribonucleic acid
SM(o)	Sample Matrix of a fuzzy object o
SOMs	Self Organizing Maps
T-SM(o)	Time based Sample Matrix of a fuzzy object o

1. INTRODUCTION

A microarray is a solid surface to which DNA is immobilised upon; each fragment of DNA fixed on to the microarray surface is called a probe and is designed to be uniquely complementary to the RNA/DNA target present in a given sample of interest. The RNA/DNA target is labelled with fluorophores and is then hybridised to the array. Microarrays are capable of simultaneously measuring the expression levels for thousands of genes. They provide a large quantity of information about an organism/cell/tissue –whether it be mutational studies (monitoring the effects of gene expression by knocking out/in a particular gene), conditional (monitoring the effects on gene expression when presenting the organism/cell to a particular environment/stress) and/or comparative (compare the presence/absence of genes in different organisms/strains). The advantage of studying many genes and many transcripts at the same time is that it provides a systematic view of how organisms/cells/tissues react in response to certain stimuli, or a global view of genome organisation (presence/absence of genes). Since microarray experiments are associated with many sources of experimental and biological variation, the resulting gene expression data is therefore noisy.

In our thesis study, we propose an efficient evolutionary clustering and prediction model for gene expression time series data which is achieved by microarray experiments. Clustering gene expression data has several limitations. One of them is that, due to the nature of microarray experiments, the produced gene expression data is fuzzy. Thus, the clusters may be diffuse and interpenetrating. In other words, for the same database, different cluster structures may be obtained by different clustering algorithms. The other limitation is that the conventional clustering approaches deal with instantaneous similarities among gene expressions, but time-based evolutionary patterns are not taken into consideration.

Several clustering techniques have been widely used for gene expression clustering. The most commonly used one is hierarchical clustering. Basically, the idea behind this approach is to build a binary tree by merging similar groups of points successively.

The main drawbacks of this approach are lack of robustness, non-uniqueness, complicated hierarchy interpretation due to inversion problems and local decision based grouping without clustering reevaluation capability [1]. The other widely used clustering technique in grouping gene expressions is density-based clustering approach. In density-based clustering algorithms, the essential task is to discover high-density regions that are separated by low-density regions in a data space. This approach is more robust than the other approaches against noisy and diffused data. Thus, unambiguous clustering performance is better than the other approaches. Because of this feature, we choose density-based clustering as clustering approach, in our study.

There are several density-based clustering algorithms such as K-Means [2], Self-organizing maps (SOMs) [3] and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [4]. One of the most popular one used for gene expression clustering is SOMs algorithm. In order to observe the pattern interpretation performance of SOMs, Tamayo *et al.* [1] develop a computer package, named as *GENECLUSER*. According to the execution results, they claim that SOMs are well suited for exploratory data analysis. Similarly, Fang *et al.* [5] develop a computer package, *supraHex*, which is an implementation of a derivation of SOMs algorithm, for training, analyzing and visualizing omics data. In spite of its popularity, SOMs approach has two major drawbacks. One of them is inaccurate clustering in case of noisy and missing data, and the other is high computational cost. Since missing and noisy data are the essential problems of microarray experiments, with this perspective, SOMs is not an appropriate solution for gene expression data. In [6], it is shown that DBSCAN produce correct set of clusters and is more robust to noisy data when compared to SOMs and k-means algorithms. Since gene expression data is uncertain, handling only noisy data problem is not sufficient. For this purpose, *FDBSCAN* (Fuzzy DBSCAN) algorithm [7], which clusters uncertain data via density-based clustering approach, is proposed by Kriegel.

In our study, we customize the uncertain data clustering method of *FBSCAN*. After deciding on the clustering approach , we develop an uncertain data clustering algorithm *M-FDBSCAN* [8] which is devised for multi-core systems. To manage uncertainty, we represent a fuzzy object with a set of sample data which is derived by a

probability density function. Increasing the cardinality of a sample data set improves the performance of uncertain data management. But, since the data size enlarges with respect to cardinality of the set, the time performance of data processing decreases. Unfortunately, even if processing huge amounts of data efficiently is a challenging problem, it is not considered in most of the uncertain data processing algorithms. Customizing the current algorithms for parallel execution is an effective method to speed up uncertain data processing. *M-FDBSCAN* is one of the first on adapting uncertain data clustering algorithms for multi-core systems. We also prove that the algorithm demonstrates a dramatic performance improvement for single-core systems.

After developing a time-efficient uncertain data clustering algorithm, in the second part of our study we develop an evolutionary clustering algorithm for gene expression time series data domain. For this purpose, we customize *M-FDBSCAN* and name the new algorithm as *E-MFDBSCAN* which is the abbrivaiation of "Evolutinary *M-FDBSCAN*". As an evolutionary clustering algorithm, *E-MFDBSCAN* generates global clusters with time series data. In the last part of our study, we propose a prediction model. Using the similarity values extracted from the patterns of global clusters, an autoregressive time series prediction model is constructed.

According to the composition of the report in order of section numbers is as below. In Chapter 2, background information about protein functions, gene expression data and DNA microarray experiments is given. In the same section, gene expression data processing approaches are also mentioned.

In Chapter 3, the approaches, methods and algorithms which we utilize in our study are described. For this purpose, in Section 3.1, general concepts of uncertain data management and mining are summarized. Density-Based clustering approach and the *FDBSCAN* algorithm is explained in Sections 3.2 and 3.3. In Section 3.4, general information about clustering gene expression time series data is given.

In Chapter 4, the thesis study is explained in detail. The *M-FDBSCAN* algorithm is explained in Section 4.1. *E-MFDBSCAN* algorithm and the proposed prediction

model are explained in Section 4.2.

Finally, the conclusion of the study is given in Chapter 5.

2. BACKGROUND

In this chapter, the biological concepts relevant to our study is described briefly. The proposed clustering and prediction solutions in this study are applied on gene expression data domain. For this purpose, before the explanation of the proposed solutions, general information about protein funtions, protein sequences, gene expression data and current experimental techniques and data processing approaches and tools are given below.

2.1. Proteins

Proteins are macromolecules that serve as building blocks and functional components of a cell, and account for the second largest fraction of the cellular weight after water. Proteins are responsible for some of the most important functions in an organism, such as constitution of the organs (structural proteins), the catalysis of biochemical reactions necessary for metabolism (enzymes), and the maintenance of the cellular environment (transmembrane proteins). Thus, proteins are the most essential and versatile macromolecules of life, and the knowledge of their functions is a crucial link in the development of new drugs, better crops.

The early approaches to predicting protein function were experimental and usually focused on a specific target gene or protein, or a small set of proteins forming natural groups such as protein complexes. These approaches included gene knockout, targeted mutations and the inhibition of gene expression [9]. However, irrespective of the details, these approaches are low-throughput because of the huge experimental and human effort required in analyzing a single gene or protein. As a result, even large-scale experimental annotation initiatives, such as the EUROFAN project [10], are inadequate for annotating a non-trivial fraction of the proteins that are becoming available due to rapid advances in genome sequencing technology. This has resulted in a continually expanding sequence-function gap for the discovered proteins [11].

In an attempt to close this gap, numerous high-throughput experimental procedures have been invented to investigate the mechanisms leading to the accomplishment of a protein's function. These procedures have generated a wide variety of useful data that ranges from simple protein sequences to complex high-throughput data, such as gene expression data sets and protein interaction networks. These data offer different types of insights into a protein's function and related concepts. For instance, protein interaction data shows which proteins come together to perform a particular function, while three-dimensional structure of a protein determines the precise sites to which the interacting protein binds itself.

Following the success of computational approaches in solving important problems such as sequence alignment and comparison [12], and genome fragment assembly [13], and given the importance of protein function, numerous computational techniques have also been proposed for predicting protein function. Early approaches used sequence similarity tools such as BLAST to transfer functional annotation from the most similar proteins. Subsequently, several other approaches have been proposed that utilize other types of biological data for computational protein function prediction, such as gene expression data, protein interaction networks and phylogenetic profiles.

2.1.1. Protein Function

The concept of protein function is highly context-sensitive and not very well defined. In fact, this concept acts typically as an umbrella term for all types of activities that a protein is involved in, be it cellular, molecular or physiological. One such categorization of the types of functions, a protein can perform, has been suggested by Bork [14].

Molecular function: The biochemical functions performed by a protein, such as ligand binding, catalysis of biochemical reactions and conformational changes.

Cellular function: Many proteins come together to perform complex physiological functions, such as operation of metabolic pathways and signal transduction, to keep

the various components of the organism working well.

Phenotypic function: The integration of the physiological subsystems, consisting of various proteins performing their cellular functions, and the interaction of this integrated system with environmental stimuli determines the phenotypic properties and behavior of the organism.

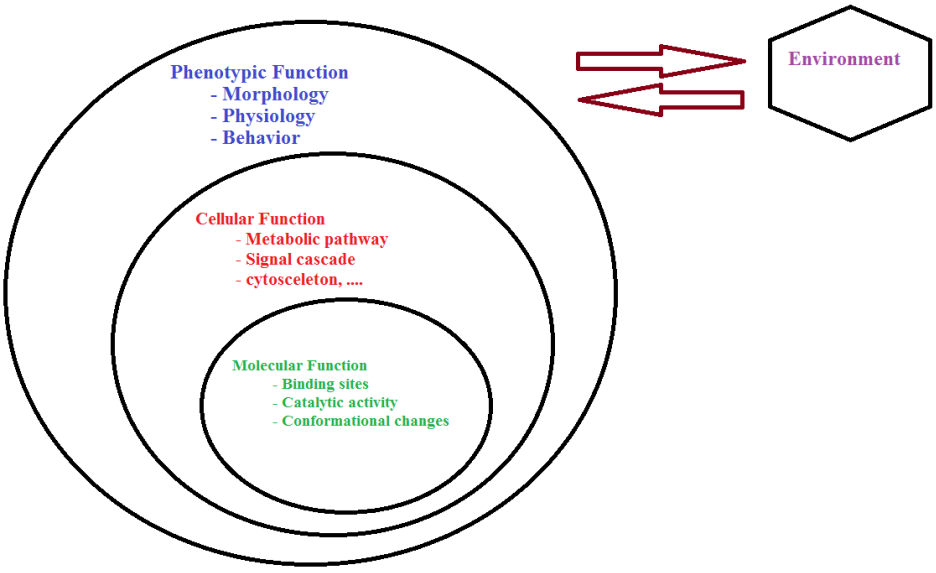

Figure 2.1. A possible hierarchical organization of the categories of protein function.

Clearly these three categories are not independent, but rather are hierarchically related as shown in Figure 2.1. Also, this is not the only categorization that has been proposed. For instance, the Gene Ontology classification scheme categorizes protein function into cellular component, molecular function and biological process.

2.1.2. Protein Sequences

The central dogma of molecular biology is the conversion of a gene to protein via the transcription and translation phases, as shown in Figure 2.2. The result of this process is a sequence constructed from twenty amino acids, and is known as the protein's primary structure. This sequence is the most fundamental form of information

available about the protein since it determines different characteristics of the protein such as its sub-cellular localization, structure and function. Even though a sequence

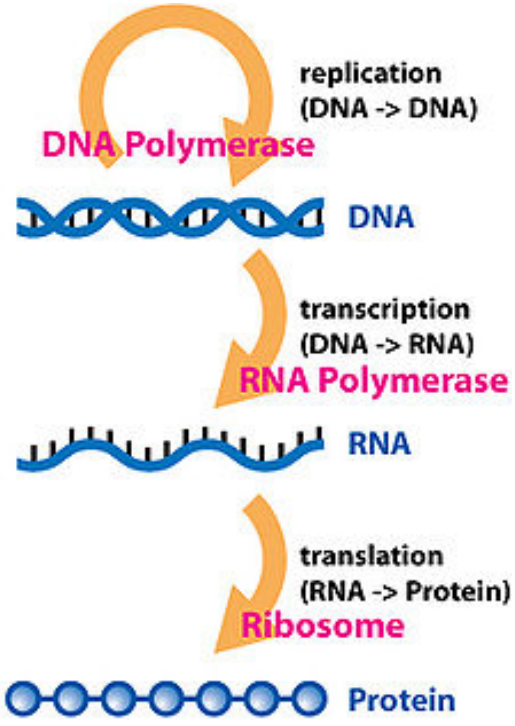

Figure 2.2. The central dogma of molecular biology: conversion of gene to protein via mRNA.

forms a unique characterization of protein, it is still a weak representation for complex operations such as prediction of its function. In comparison, more complex forms of data such as gene expression and protein interaction networks offer a deeper insight into the mechanisms leading to the performance of a protein's function, and are thus more useful for predicting function.

2.2. Gene Expression Data

Protein synthesis from genes occurs in prokaryotic organisms in two phases as shown in Figure 2.2. In the transcription phase, an mRNA is created from the original gene by converting the latter to the corresponding RNA code. The protein is then synthesized from mRNA by translating the RNA code to the corresponding amino acid sequence according to the codon translation rules.

Gene expression experiments are a method to quantitatively measure the transcription phase of protein synthesis Nguyen *et al.* [15]. The most common category of these experiments uses square-shaped glass chips measuring as little as 1 inch on either side, also known as cDNA microarrays, and hence the alternate name microarray experiments. The experiment is carried out in the following stages. In the first stage, the chip is laid out with a matrix of dots of cDNAs, usually several thousands in number, one corresponding to each of the gene being measured. In parallel, mRNA is extracted from both the normal as well as the cells of the organism that have been exposed to the condition being studied. Next, these mRNA are reverse-transcripted to cDNA and colored with green and red colors respectively. These colored cDNAs are then spread on the microarray chip, leading to a hybridization of the cDNA already on the chip with those produced by the genes in the two types of cells. This generates a spot of a certain color on the chip for each gene, which denotes its expression level. In the final stage of the experiment, the intensity of this region is measured by a laser scanner, connected to a computer, which generates a real valued measurement of the expression of each gene as the ratio of the log intensities of red and blue colors in the region. The result of the experiment thus is a measurement of the transcription activity of the genes under the specified condition. A detailed illustration of this procedure is shown in Figure 2.3.

The primary advantage of gene expression experiments are that they offer an effective method for observing the simultaneous activity of thousands of genes under a given experimental condition. Using these activity measurements, several important inferences can be drawn about the underlying biological phenomena, such as the active pathways under the given condition. This ability to observe a global pattern of activity of genes, particularly when observed over several multiple related experimental conditions, has motivated the use of microarrays for a variety of biological studies [16]. Also, since data generated from one experiment can be useful for several other studies, several repositories have been set up in order to make such data publicly accessible.

Usually, the format of gene expression data is very simple, i.e., a rectengular matrix, in which the rows correspond to genes, the columns to conditions, and the

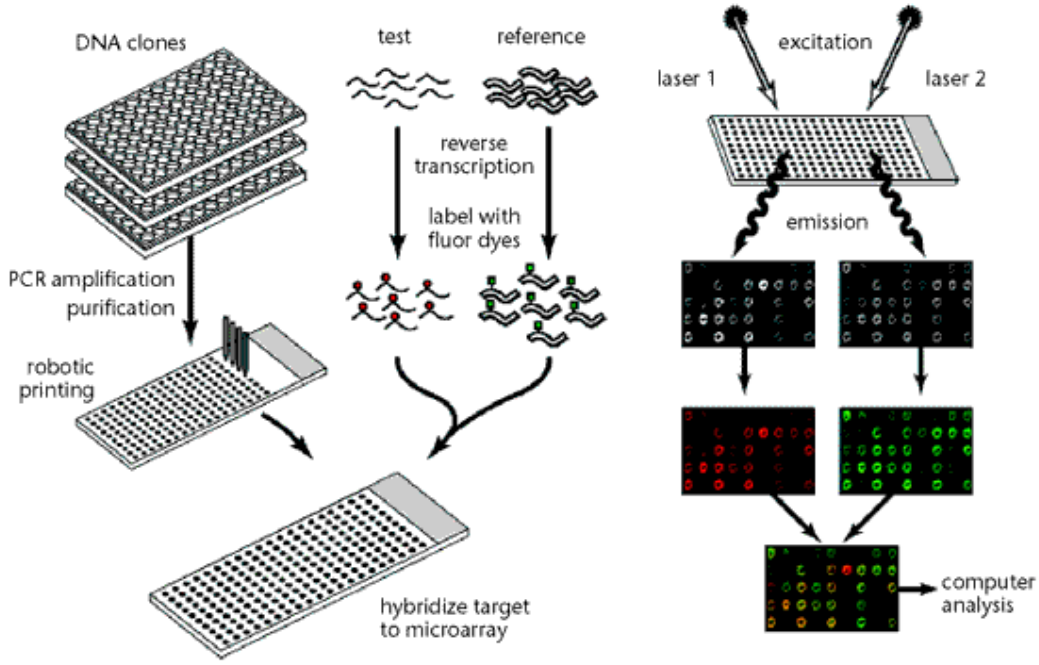

Figure 2.3. An illustration of the microarray experimental procedure.

entries denote the expression measurement of a gene under a particular condition. However, since the data is generated experimentally, there may be several phenomena that may affect the quality of data produced by an experiment. Some such problems are varying degrees of hybridization across the chip, background noise in the images produced, and a difference of scale between the different experiments constituting a microarray data set. Several statistical methods have been developed for addressing these problems [17], which use the information in the experimental design, as well as the data generated, in order to reduce the effect of these factors in the processed data set. Another important factor to consider in microarray data analysis is that the data used in research is generally of two kinds: static and temporal [18]. The first category consists of data sets containing snapshots of the expression of certain genes in different samples under the same conditions, while the latter, also known as time-series gene expression data [19], consists of data sets capturing the expression of certain genes of the same organism at different instances of time. It is important to consider these characteristics of the data when it is used for computational analysis, such as running analysis algorithms to infer protein function from gene expression data.

Early approaches identified functional associations between genes by measuring the similarity between their expression profiles using statistical methods. In a study focused on identifying novel genes which may contribute to prostate cancer in humans [20], 40,000 genes were examined for co-expression with five genes known to be associated with prostate cancer using the Guilt by Association (GBA) principle. As a result, eight novel genes that are significantly co-expressed with at least one known prostate cancer causing gene are identified and are verified as being related to processes leading to the disease. However, these studies were usually very narrow in scope, and involved significant human intervention in identifying the seed or the target genes. This allowed the application of more generic techniques from data mining for this task. These can be grouped into the following three categories:

Clustering-based approaches: An underlying hypothesis of gene expression analysis is that functionally similar genes have similar expression profiles, since they are expected to be activated and repressed under the same conditions. Because clustering is a natural approach for grouping similar data points, approaches in this category cluster genes on the basis of their gene expression profiles, and assign functions to the non-annotated proteins using the most dominant function for the respective clusters containing them.

Classification-based approaches: A more direct solution to the problem of predicting protein function from gene expression profiles is the data mining approach of classification. Thus, approaches in this category build various types of models for expression-function mapping using classifiers, such as neural networks, SVMs and the naive Bayes classifier, and the use these models to annotate novel proteins.

Temporal analysis-based approaches: Temporal gene expression experiments measure the activity of genes at different instances of time, for instance, during a disease. This behavior can also be used to predict the protein function. Thus, approaches in this category derive features from this temporal data and use classification techniques to predict the functions of non-annotated proteins.

3. RELATED APPROACHES AND ALGORITHMS

In our study, we utilize several approaches and algorithms to establish the proposed methodology. In this section, we mention about these approaches and algorithms. Since, gene expression data is uncertain data, in some part of our study we propose some solutions about managing this uncertainty. Thus, in the first part of the section, we give short information about uncertain data management and mining concepts. On the other hand, the developed clustering algorithms, *M-FDBSCAN* and *E-MFDBSCAN* are density-based uncertain data clustering algorithms. Thus, in the second part of the section we briefly explain the idea behind density-based uncertain data clustering approach and give the descriptions of several algorithms developed according to this approach. *FDBSCAN* algorithm is in the focus of the next part of the section. An elaborated description of the algorithm is given in this part. Since our proposed algorithms *M-FDBSCAN* and *E-MFDBSCAN* are the variants of *FDBSCAN*, this algirotihm has an important place in our study. In the last part of the section, general concepts of clustering of gene expression time series data are given.

3.1. Uncertain Data Management and Mining

Uncertain data management has been a revival in interest in recent years because of a number of new fields which utilize this kind of data. For example, in fields such as privacy preserving data mining, additional errors may be added to data in order to mask the identity of the records. Often the data may be imputed using statistical methods such as forecasting. In such cases, the data is uncertain in nature. Such data sets may often be probabilistic in nature. In other cases, databases may show *existential uncertainty* in which one or more records may be present or absent from the data set. Such data sets lead to a number of unique challenges in processing and managing the underlying data. The field of uncertain data management presents a number of challenges in terms of collecting, modeling, representing, querying, indexing and mining the data. Many of these issues are inter-related and cannot easily be addressed independently. While many of these issues have been addressed in recent research, the

research in this area is often quite varied in its scope. For example, even the underlying assumptions of uncertainty are different across different papers. It is often difficult for researchers to find a single place containing a coherent discussion on the topic. The essential research issues of uncertain data can be grouped as below:

- *Modeling and System Design for Uncertain Data:* The nature of complexity captured by the uncertain data representation relies on the model used in order to capture it. The most general model for uncertain data is the *possible worlds model* [21], which tries to capture all the possible states of a database which are consistent with a given schema. The generality of the underlying scheme provides the power of the model. On the other hand, it is often difficult to leverage a very general representation for application purposes. In practice, a variety of simplifying assumptions (independence of tuples or independence of attributes) are used in order to model the behavior of the uncertain data. On the other hand, more sophisticated techniques such as probabilistic graphical models can be used in order to model complex dependencies. This is a natural tradeoff between representation power and utility. Furthermore, the design of the system used for representing, querying and manipulating uncertain data critically depends upon the model used for representation.
- *Management of Uncertain Data:* The process of managing uncertain data is much more complicated than that for traditional databases. This is because the uncertainty information needs to be represented in a form which is easy to process and query. Different models for uncertain data provide different tradeoffs between usability and expressiveness. Clearly, the best model to use depends upon the application at hand. Furthermore, effective query languages need to be designed for uncertain data and index structures need to be constructed. Most data management operations, such as indexing, join processing or query processing need to be fundamentally re-designed.
- *Mining Uncertain Data:* The uncertainty information in the data is useful information which can be leveraged in order to improve the quality of the underlying results. For example, in a classification application, a feature with greater uncer-

tainty may not be as important as one which has a lower amount of uncertainty. Many traditional applications such as classification, clustering, and frequent pattern mining may need to re-designed in order to take the uncertainty into account.

3.2. Density-Based Uncertain Data Clustering

The presence of uncertainty changes the nature of the underlying clusters, since it affects the distance function computations between different data points. A technique has been proposed in [7] in order to find density-based clusters from uncertain data.

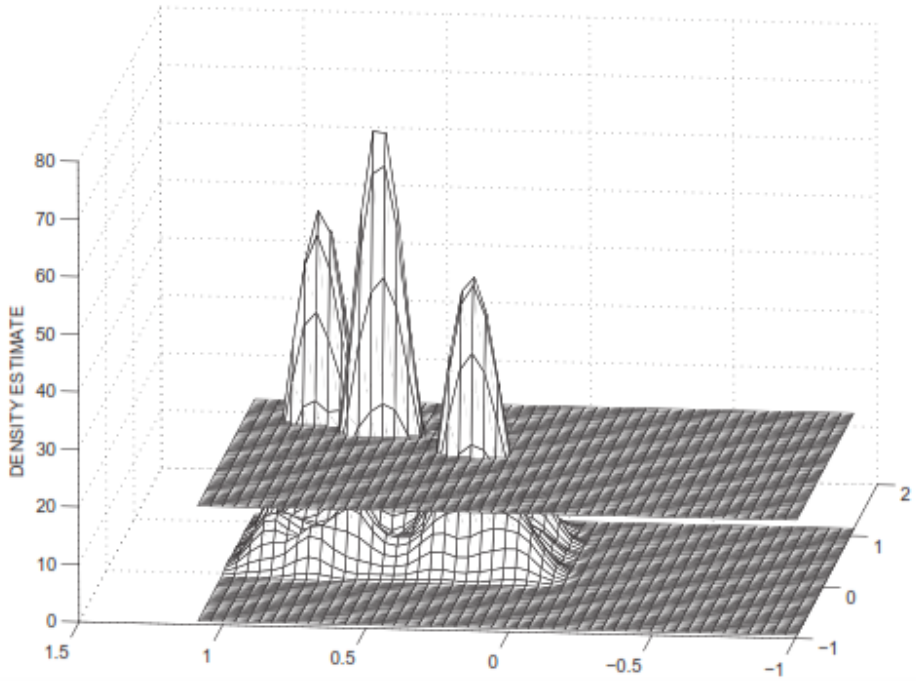

Figure 3.1. Density-Based profile with lower density threshold.

The key idea in this approach is to compute uncertain distances effectively between objects which are probabilistically specified. The fuzzy distance is defined in terms of the distance distribution function. This distance distribution function encodes the probability that the distances between two uncertain objects lie within a certain user-defined range. Let $d(X, Y)$ be the random variable representing the distance between X and Y. The distance distribution function is formally defined as follows:

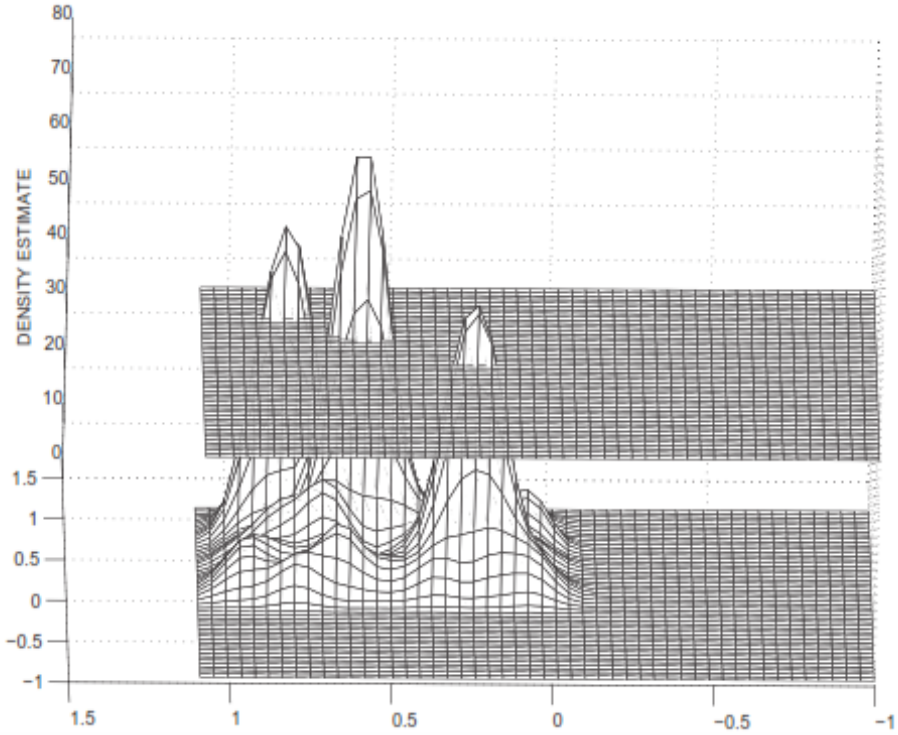

Figure 3.2. Density-Based profile with higher density threshold.

Definition: Let X and Y be two uncertain records, and let *p(X,Y)* represent the distance density function between these objects. Then, the probability that the distance lies within the range (a, b) is given by the following relationship, 3.1:

$$P(a \leqslant d(X,Y) \leqslant b) = \int_a^b p(X,Y)(z)dz \tag{3.1}$$

Based on this technique and the distance density function, the method in [7] defines a *reachability probability* between two data points. This defines the probability that one data point is directly reachable from another with the use of a path, such that each point on it has density greater than a particular threshold. The data points are grouped into clusters when they are reachable from one another by a path which is such that every point on this path has a minimum threshold data density. To this effect, the algorithm uses the condition that the e-neighborhood of a data point should contain at least *MinPts* data points. The algorithm starts off at a given data point

and checks if the μ-neighborhood contains *MinPts* data points. If this is the case, the algorithm repeats the process for each point in this cluster and keeps adding points until no more points can be added. One can plot the density profile of a data set by plotting the number of data points in the μ-neighborhood of various regions, and plotting a smoothed version of the curve. This is similar to the concept of probabilistic density estimation. Intuitively, this approach corresponds to the continuous contours of intersection between the density thresholds of Figures 3.1 and 3.2 with the corresponding density profiles. The density threshold depends upon the value of *MinPts.* As it is noted, the data points in any contiguous region will have density greater than the threshold. Also it is noted that the use of a higher density threshold (Figure 3.2) results in 3 clusters, whereas the use of a lower density threshold results in 2 clusters. In the following section, *FDBSCAN*, is explained in detail.

Another related technique discussed in [22] is that of hierarchical density based clustering. An effective (deterministic) density-based hierarchical clustering algorithm is OPTICS [23]. The core idea in OPTICS is quite similar to DBSCAN and is based on the concept of *reachability distance* between data points. While the method in DBSCAN defines a *global density parameter* which is used as a threshold in order to define reachability, the work in [22] points out that different regions in the data may have different data density, as a result of which it may not be possible to define the clusters effectively with a single density parameter. Rather, many different values of the density parameter define different (hierarchical) insights about the underlying clusters. The goal is to define an implicit output in terms of ordering data points, so that when the DBSCAN is applied with this ordering, once can obtain the hierarchical clustering at any level for different values of the density parameter. The key is to ensure that the clusters at different levels of the hierarchy are consistent with one another. As a result of an observation, the clusters defined over a lower value of ε are completely contained in clusters defined over a higher value of ε, if the value of *MinPts* is not varied. Therefore, the data points are ordered based on the value of ε required in order to obtain *MinPts* in the ε-neighborhood. If the data points with smaller values of ε are processed first, then it is assured that higher density regions are always processed before lower density regions. This ensures that if the DBSCAN algorithm is used for

different values of ε with this ordering, then a consistent result is obtained. Thus, the output of the the data points are processed. Since the OPTICS algorithm shares so many characteristics with the DBSCAN algorithm, it is fairly easy to extend the used for extending the DBSCAN algorithm. This is referred to as the FOPTICS algorithm. One of the core-concepts needed to order to data points is to determine the value of ε which is needed in order to obtain *MinPts* in the corresponding neighborhood. In the uncertain case, this value is defined probabilistically, and the corresponding expected values are used to order the data points.

3.3. FDBSCAN

FDBSCAN [7] is a density-based uncertain data clustering algorithm which is an extension of DBSCAN [4]. The proposed algorithms *M-FDBSCAN* and *E-MFDBSCAN*, in our study, are based on the uncertain data clustering method of *FDBSCAN*. Thus, before the elaborated explanations of the two algorithms, we present the proposed uncertain data clustering solution of of *FDBSCAN*. For the sake of understandability, the relevant definitions used in the algorithm are given below.

3.3.1. Definitions

Definition 3.1. *Object o is called a* core object *w.r.t. ε and μ in a set of objects D, if $|N_\varepsilon(o)| \geqslant \mu$, where $N_\varepsilon(o)$ denotes the subset of D contained in the ε-neighborhood of o.*

Definition 3.2. *Object p is* directly density-reachable *from object o w.r.t. ε and μ in a set of objects D, if o is a core object and $p \in N_\varepsilon(o)$, where again $N_\varepsilon(o)$ denotes the subset of D contained in the μ-neighborhood of o.*

Definition 3.3. *An object p is* density-reachable *from object o w.r.t. ε and μ in a set of objects D, if there is a chain of objects $p_1,\ldots,p_n$, $p_1 = o$, $p_n = p$ such that $p_i \in D$ and p_{i+1} is directly density-reachable from p_i w.r.t. ε and μ. Object p is* density-connected *to object q w.r.t. ε and μ in the set of objects D, if there is an object $o \in D$ such that both p and q are density-reachable from o w.r.t. ε and μ in D.*

Definition 3.4. *Let d: $D \times D \rightarrow IR_0^+$ be a distance function, and let P(d(o,o') $\leqslant$ b) denote the probability that d(o,o') is smaller than b. Then a probability distribution function $P_d : O \times O \rightarrow (IR_0^+ \rightarrow [0..1])$ is called* a distance distribution function *if the following condition holds:*

$$P_d(o, o')(b) = P(d(o, o') \leqslant \ b) \tag{3.2}$$

Definition 3.5. *Let D be a database, and let $P_d : D \times D \rightarrow (IR_0^+ \rightarrow [0..1])$ be a distance distribution function. Then, the* core object probability *of an object o is defined as :*

$$P^{core}_{\varepsilon,\mu,d,D}(o) = \sum_{\substack{A \subseteq D \\ |A| \geq \mu}} \prod_{p \varepsilon A} P_d(p, o)(\varepsilon) \prod_{p' \varepsilon D|A} (1 - P_d(p', o)(\varepsilon)) \tag{3.3}$$

Definition 3.6. *Let D be a database, and let P_d:$D \times D \rightarrow (IR_0^+ \rightarrow [0..1])$ be a distance distribution function. Then, the* reachability probability *of an object p w.r.t. o is defined as follows:*

$$P^{reach}_{\varepsilon,\mu,d,D}(p, o) = P^{core}_{\varepsilon,\mu-1,d,D|\{p\}}(o) \bullet \ P_d(p,o)(\varepsilon) \tag{3.4}$$

3.3.2. The Algorithm

Core object concept is first proposed by DBSCAN and an enhanced version of it is also used in *FDBSCAN*. The *core object probability* of an object o indicates the likelihood that o is a core object. Based on the core object probability definition, it can be defined how likely it is that an object p is directly density reachable from an object o. In the traditional density-based clustering approach, two conditions have to hold. First, o has to be a core object, and second, the distance between p and o has to be smaller than or equal to ε. According to *FDBSCAN* algorithm, both of these conditions are fuzzy, holding only with a certain probability. An object p is added to the current cluster, if $P^{reach}_{\varepsilon,\mu,d,D}(p, o) \geqslant 0.5$ where o is the current query object. Since, by definition $P_d(p,o)(\varepsilon) \leqslant 1$, if $P^{core}_{\varepsilon,\mu-1,d,D|\{p\}}(o) < 0.5$ holds, for any object p, $P^{reach}_{\varepsilon,\mu,d,D}(p,o)$

can never reach to 0.5. Therefore p will not be added to the current cluster. The other problem is how to compute the *reachability probability* efficiently. For this purpose, *FDBSCAN* proposed a generally applicable approach based on monte-carlo sampling. If the fuzzy object is described by a continuous probability density function, it can be easily sampled according to this function and derived thus a sequence of samples. *FDBSCAN* assumes that each fuzzy object $\boldsymbol{x}$ is represented by a sequence of $\boldsymbol{s}$ sample points, i.e. $\boldsymbol{x}$ is represented by $\boldsymbol{s}$ different representation $<\boldsymbol{x_1},..,\boldsymbol{x_s}>$. Based on the sample sequences discrete density functions consisting of $\boldsymbol{s^2}$ many discrete distance values can be computed. The core object and reachability probabilities are determined with using these $\boldsymbol{s^2}$ meaningful samples.

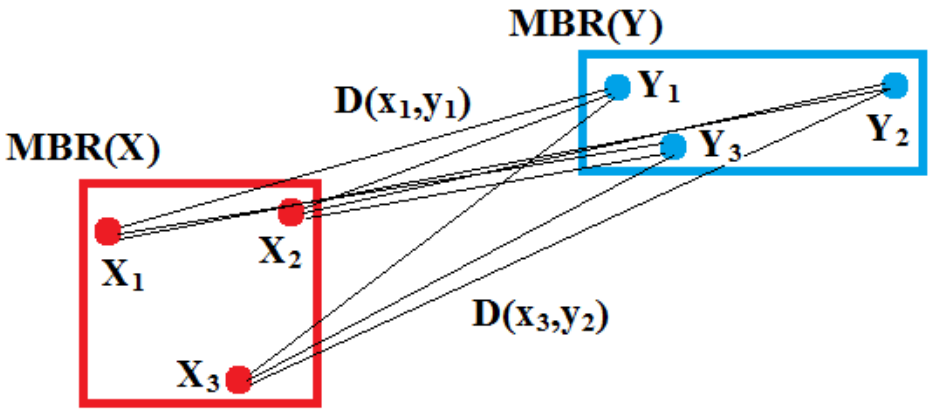

Figure 3.3. Minimum Bounding Rectangles for X and Y fuzzy objects where s=3.

As it is illustratred in Figure 3.3, *FDBSCAN* propose a geometrical representation type, called *minimum bounding rectangle, MBR,* to represent the fuzzy objects with their sample points in 2D-space. For each fuzzy object $\boldsymbol{x}$ in D, $MBR(\boldsymbol{x})$ is established, before the clustering issues. Since the distance values between the sample points at the boundaries of the rectangles are inclusive and sufficient, *MBR* structure improves the computation performance by means of minimizing the number of distance calculations.

The other essential data structure proposed by *FDBSCAN* is *sample* matrix, which is used for the computation of $P^{core}_{\varepsilon,\mu-1,d,D|\{p\}}(o)$. *Sample matrix* is a $s \times s$ matrix, where s is the number of derived sample points for a fuzzy (uncertain) data

object o. In Figure 3.4, a sample 4×4 *sample matrix* SM(o) for fuzzy data object$o$, is shown. Each row o_i in *sample matrix* represents the derived sample points of o, where $\{o_i|<o_1,\ldots,o_i,\ldots,o_s>\}$. Each column D_j represents the jth database instance, where $\{x_j| < x_1,\ldots,x_i,\ldots,x_s > \in D \wedge x_j \neq o_j\} \cup o_i$. Finally, each matrix element is $m_{i,j} = |N_{\varepsilon,Dj}(o_i)|$, where $N_{\varepsilon,Dj}(o_i)$ denotes the set $\{x_j/d(o_i,x_j)\leqslant \varepsilon \wedge x_j \in D_j\}$ and $d(.,.)$ denotes any distance function.

SM(o)

$$
\begin{array}{c} \\ o1 \\ o2 \\ o3 \\ o4 \end{array}
\begin{array}{c}
\begin{array}{cccc} \mathbf{D1} & \mathbf{D2} & \mathbf{D3} & \mathbf{D4} \end{array} \\
\begin{bmatrix} 4 & 7 & 5 & 9 \\ 6 & 11 & 8 & 5 \\ 7 & 3 & 9 & 7 \\ 2 & 0 & 5 & 6 \end{bmatrix}_{s=4}
\end{array}
$$

Figure 3.4. A 4×4 sample matrix.

To decide whether it is required or not to calculate the distance between two sample points o_i and x_j, the rectangles $MBR(o)$ and $MBR(x)$ are used. By considering the locations of $MBR(o)$ and $MBR(x)$; i) if $d_{max}(o,x) \leqslant \varepsilon$ holds, $m_{i,j}$ for all i and j is increased by 1 or ii) if $d_{min}(o,x) > \varepsilon$ holds, none of $m_{i,j}$ is increased or iii) if $d_{min}(o,x) \leqslant \mu \leqslant d_{max}(o,x)$ then all the distances $d(o_i,x_j)$ must be calculated. After constructing the sample matrix SM(o), the reachability probability values of all objects x in D, w.r.t. o, of which the usage is mentioned above, can be computed according to the given instruction in [7].

3.4. Clustering of Gene Expression Time-Series Data

In the gene expression context, clustering is used to identify subsets of genes that behave similarly along time under the set of test conditions; that is, to cluster gene

expression time-series data. In clustering, as in any data analysis, issues ranging from problem definition to a critical diagnosis of the results must be addressed.

3.4.1. Time-series

A time-series is often defined as a series of values of a variable taken in successive periods of time. The variables come from a variety of different domains from engineering to scientific research, finance and medicine. The range, noise, scaling and shifting factors of the values that such variables can take depend on the nature of the variables and the instrument utilised to measure them. The instants in time at which the measurements are taken are known as time points. The length between time points can vary or be constant and is called sampling interval. There is a well-established area in statistical analysis of data dedicated to the study of time-series. The statistical analysis of time-series accounts for the fact that data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for. In general most of the analysis is focused towards uni-variate time-series with a large number of measurements and equally distant time points.

There are many methods to model time-series. The selection of the appropriate technique will depend on the application and the user's preference.

An example of a model is the auto regressive (AR) model:

$$x_t = \delta + \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \ldots + \varphi_p x_{t-p} + A_{t-2} \tag{3.5}$$

where x_t is the time-series, A_t is noise, and

$$\delta = (1 - \sum_{i=1}^{p} \varphi_i)\mu \tag{3.6}$$

where μ is the process mean. An autoregressive model is a linear regression of the current value of the series against one or more prior values of the series. The value of

p is called the order of the AR model. In our proposed prediction model, we also use this model for our prediction model.

One of the important properties of the time-series for their statistical analysis is *autocorrelation*. *Autocorrelation* refers to the correlation of a time-series with its own past and future values. It is the correlation of the time-series with itself but shifted in time k time points, k is usually called the lag. Autocorrelation complicates the application of statistical tests by reducing the effective sample size. There are several tools for assessing the autocorrelation: time-series plot, lagged scatter plot and autocorrelation function.

3.4.2. Gene expression time-series

Gene expression time-series have two main characteristics, they are very short (i.e., four to twenty samples) and are usually unevenly sampled. The existing literature on short time-series focuses primarily on testing and estimating autocorrelation. Samples under 50 observations are already considered too short for a classical statistical analysis. Calculation of the autocorrelation at different lags is an essential instrument to identify the dependency structure of the series but the estimation of the autocorrelation is biased with small samples, [24] and [25].

3.4.3. Similarity of gene expression time-series

Similarity is understood as the resemblance, likeness, or equivalence of two objects. It is a relative term, which only makes sense when comparing more than two elements or when a threshold is utilised. Considering three different objects A, B and C, there are three possibilities for the similarity of A. A can be more similar to B than to C, or more similar to C than to B, or as similar to C as to B. In order to be able to asses the similarity, a quantitative measure of likeness has to be utilised. It is a common practice to use correlation or distance metrics to quantify such resemblance. The suitability and performance of a similarity measure for a specific comparison depend on the nature of the objects to compare. Therefore, the requirements of similarity of gene

expression time-series have to be considered in order to design or select the appropriate similarity measure.

3.4.3.1. Similarity requirements for co-expression. The general objective of the clustering of gene expression data is the identification of co-expressed genes. However, there is not a clear definition of co-expression in the literature. In general, it is understood that co-expressed genes have similar pat-terns of expression. The common idea behind the concept of similar expression patterns lies in the direction of change of the expression level across time points. Co-expression has not a common and well defined meaning in the literature, which leaves a wide open door for the suggestion for new and alternative clustering procedures.

Three basic similarity requirements have been identified; the similarity measure should be able to handle:

- Scaling and shifting problems
- Unevenly distributed sampling points
- Shape (internal structure)

Scaling and shifting problems: A promoter is a structural regulatory sequence recognised by the sigma factor of the RNA polymerase holoenzyme (the protein that is used to read the DNA for transcription). Genes that share a common sequence, will therefore share their expression, they will be switched on at the same time but not necessarily at the same level. The reason is that the recognition effciency is not the same for every gene having that promoter. This is one of the situations leading to scaled and shifted expressions. Therefore, scaling and shifting factors in the expression level hide similar expressions and have to be eliminated or not considered when assessing the similarity between expression profiles. Other possible sources for scaling and shifting problems are intrinsic of the microarray experiment (e.g. label effciency of the dyes), which are usually eliminated in the normalisation procedure.

Synchronisation of biological processes is not an easy task. Common processes may unfold at different times in different experiments or individuals producing horizontal shifts in the resulting time-series. For a few number of time points the identification of horizontal shifts can possibly be made after the clusters are obtained while for longer series temporal aligning techniques can be utilised.

Unevenly distributed sampling points: Since, gene expression time series rely on samples of the actual biological process, the higher the sampling frequency the more information one has to recreate the actual process. However, it is not possible to achieve high sampling frequency in microarray experiments due mainly to the time and resources that would require. Therefore, biological processes are sampled at shorter intervals of time when intense biological activity or when the activity of interest is taking place, leading to unevenly distributed sampling points. In consequence, the length of the sampling interval is informative and should be considered in similarity comparisons.

Shape (internal structure): The main difference between a set of measurements and a time series, is the internal structure, therefore a time-series can not be treated as independent identically distributed data. The internal structure can be described by different models and in general it is reflected in the shape of the series. In microarray experiments, the intensity of gene expression is not relevant, instead, the relative change of intensity characterised by the shape of the expression profile is regarded as characteristic and informative. A necessary condition for the existence of internal structure or characteristic shape is the temporal order of measurements. Therefore, the similarity function should not allow a change in the order. The internal structure can be represented by a statistical model, by deterministic functions or by symbols describing the series.

3.4.4. Clustering of gene expression time-series

After having described the requirements of similarity for gene expression time-series, this section specifies the requirements for clustering the aforementioned se-

ries.Several requirements can be identified for the clustering algorithm, in specific it should be able to handle:

- Unknown number of clusters
- Varying membership
- Outliers
- Noise

Number of clusters: Unsupervised clustering is the most common approach for clustering gene expression data. This means that there is no previous knowledge of the number and characteristics of the clusters forming the data. Different clustering algorithms have special techniques for identifying the number of clusters. A common approach to identify the number of clusters is the use of validity measures; the data is clustered defining different number of clusters and the best value of the validity measure will identify the most convenient number. In fuzzy clustering the fuzziness of the partition can be used to evaluate the goodness of the results for different number for clusters Höppner *et al.* [26]. Other approaches do not have to define a number of clusters, because it is obtained as a result of the partition, but additional parameters have to be defined. For example, in the CAST algorithm [27], the number of clusters has not to be determined. Instead, a parameter called affnity threshold is used to determine what is the minimum similarity required between an object and a cluster for that object to be a member, and not all the genes are assigned to a cluster. In this approach each cluster is formed by alternating between adding and removing genes from the current cluster until such time that changes no longer occur or a maximum of iterations has been executed. Therefore, the number of clusters obtained from the partition of the data set depends on the selection of the affinity threshold. In general, a clustering algorithm for gene expression data should be able to identify the hidden number of clusters by appropriate means relevant to the algorithm.

Varying membership: Several researches have identified and emphasize the importance of overlapping clusters in gene expression clustering analysis, [28, 29]. The

partition of genes into classical sets implies that each gene has been associated with a single biological function or process which may be an oversimplification of the biological system. Therefore, genes should be allowed to have varying probability or membership degree to different clusters to allow connection of genes to more than one cluster, revealing distinct aspects of their function and regulation. The EM algorithm utilised for model-based clustering allows for partial credit to different clusters, that is, genes have varying probability to belong to different clusters. Other approach is fuzzy clustering, which allows genes to belong to more than one group by assigning different degrees of membership to each cluster.

Outliers: Given the restricted number of time points in gene expression time-series, an outlier has a high influence on the similarity measure. The clustering algorithm should be able to minimise this impact. A good example is [30], where the authors use the jack-knife correlation. This measure corresponds to the minimum of all the possible calculations of correlation between two series, where each calculation is done with the omission of a different time point. In this way, the effects of outliers in the clustering procedure are reduced. Another approach is the identification of outliers as a previous step to the clustering procedure.

Noise: It is well known that microarray experiments are subject to a large experimental error producing very noisy measurements. The clustering algorithm should be able to handle these common levels of noise. There are several approaches to achieve this. For example, in fuzzy clustering a noise cluster can be added to the partition [31]. This cluster attracts all those genes which do not show a minimum level of similarity to the rest of the clusters and reduces the influence of this group in the whole partition. Some algorithms based in stochastic models have remarked the possibility of incorporating noise in the model [32, 33].

4. THESIS STUDY

In our study we propose a methodology to cluster gene expression time-series data efficiently and, by the help of the evolutionary patterns of the clusters, to predict the similarity patterns of the clusters that will be generated at the next future time point. For this purpose, we develop two uncertain data clustering algorithms, *FDBSCAN* and *E-MFDBSCAN* and a prediction model. *FDBSCAN* is an algorithm devised for multi-core systems. After publishing this study, we develop an evolutionary clustering algorithm, *E-MFDBSCAN* which generates time-based global clusters. In *E-MFDBSCAN* the procedures and the methods of *FDBSCAN* that enables parallel execution are inherited. In the last step of the study, we develop a prediction model which is established on the extracted evolutionary patterns of the global clusters generated by *E-MFDBSCAN*.

In the following sections, these algorithms and the proposed prediction model are explained.

4.1. M-FDBSCAN

In many data mining applications, clustering algorithms are executed on a huge amount of uncertain data. Thus, high computational cost of uncertain data requires time-efficient algorithms. In [34], one of the most well-known clustering algorithms, k-means algorithm, is re-implemented for multi-core systems. To each core, some part of the dataset is assigned for clustering. After k-means algorithm is applied to the relevant part of the datasets on each dedicated core, simultaneously, a merge operation is run to get the final clusters. But in this study, uncertain data management is not considered. Uncertain data handling versions of k-means, DBSCAN, and OPTICS clustering algorithms are proposed in [7, 22], and [35], respectively. Based on the fuzzy c-means algorithm, novel fuzzy clustering solutions for different data domains and problems are proposed in [36–38]. But the proposed algorithms in [7, 22] and [35, 38] are devised for sequential execution.

Recent studies are either exclude data uncertainty or not devised for multi-core systems. In our thesis study, we process gene expression data which is assumed as uncertain data. Thus, we aim to develop uncertain data clustering algorithm which has parallel execution capability. For this purpose, we adapt *FDBSCAN* to multi-core systems in order to have fast processing and name this new algorithm as *M-FDBSCAN* (Multi-core *FDBSCAN*). With this perspective, *M-FDBSCAN* is one of the first on adapting uncertain data mining algorithms for multi-core systems.

In *M-FDBSCAN*, the data domain is split into c rectangular data regions, where c denotes the number of cores in the multi-core system. After clustering operations are applied to each rectangular data region concurrently, the generated set of clusters is merged to construct the final state of the clusters.

4.1.1. Computational Aspects

Sample matrix concept, which is described in Section 3.3.2, is the core concept of *FDBSCAN*. But at the same time, construction of the sample matrices for each fuzzy data object is the most time-consuming part of the algorithm. In *M-FDBSCAN*, we aim to minimize the time cost of matrix construction.For this purpose, we propose a splitting method. According to this method, a dataset is split into c sub datasets. In Figure 4.1, a 4×4 sample matrix illustration is given for a non-split dataset of N fuzzy data objects. As it is illustrated in Figure 4.2, the dataset is split into 4 sub datasets of N/4 fuzzy data objects. In *M-FDBCSAN*, for each sub dataset sample matrices are construcuted concurrently. During the construction of the sample matrices, the number of distance computations required for each sub dataset is less than the required number of distance computations without splitting. The relevant computational aspects are elaborated below.

For a dataset of f fuzzy data objects and s sample points per each fuzzy data

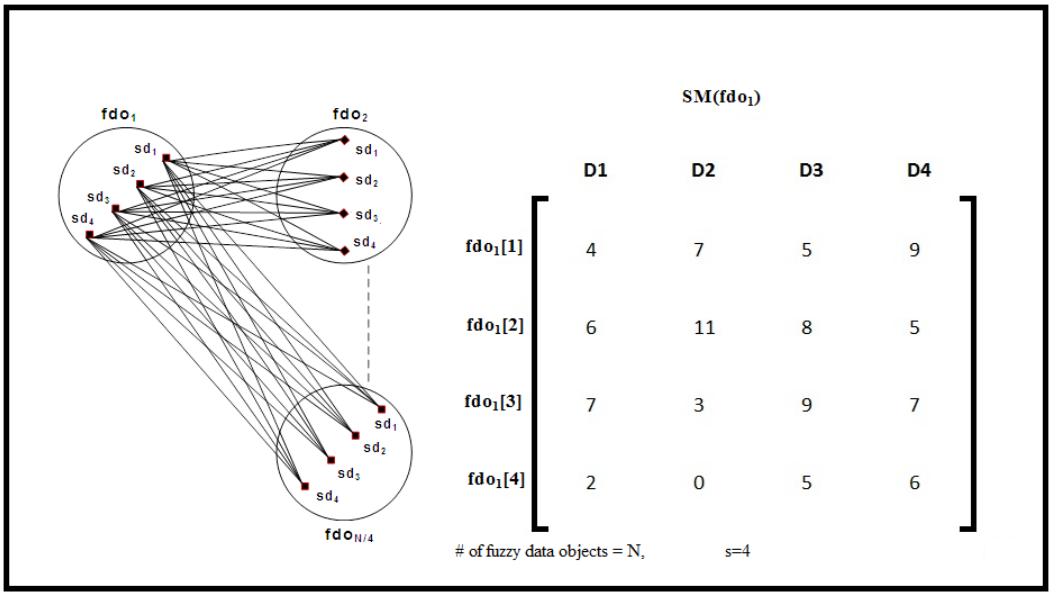

Figure 4.1. A 4×4 sample matrix illustration without splitting dataset.

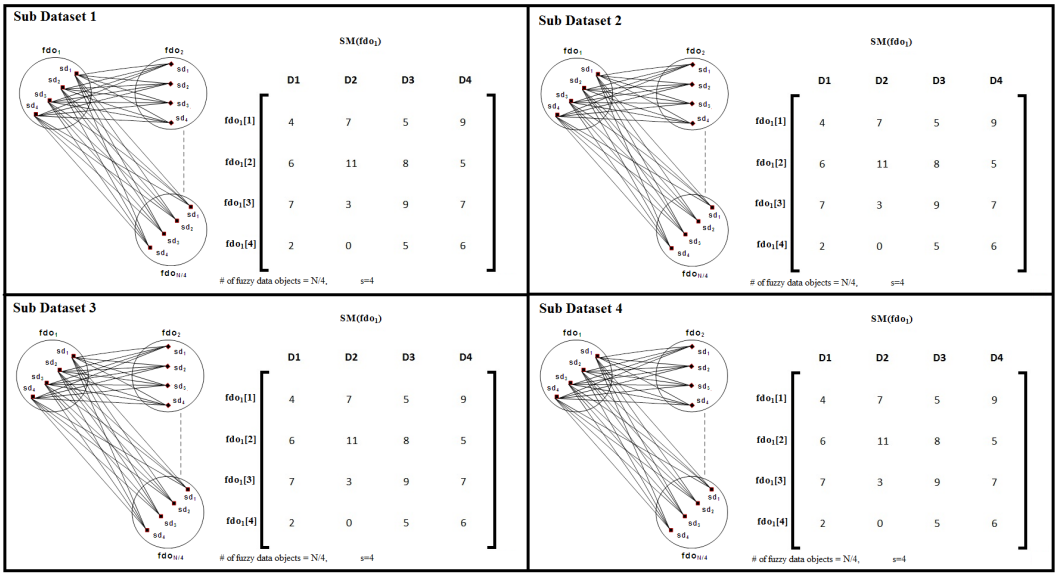

Figure 4.2. A 4×4 sample matrix illustration with splitting dataset into 4 sub datasets.

object, totally, the number of distance computaions is:

$$s^2 \times ((f\ -1) + (f-2) \ldots + 1) = s^2 \times (f \times \frac{f-1}{2}) = s^2 \times \frac{f^2 - f}{2} \tag{4.1}$$

Please note that the result is not simply $s^2 \times f^2$. This is because the distances between the fuzzy data object pairs (fdo_x, fdo_y) and (fdo_y, fdo_x) are obviously the same. Thus, trivially, only one distance computation for a certain pair is sufficient.

In our proposed algorithm, we aim to parallelize the distance computation task by splitting the dataset into c sub-datasets. Here, c is the number of cores in the associated multi-core system. Thus, assuming that the sizes of all of the sub datasets are equal, the number of distance computations done by each core:

$$s^2 \times (f^2/c^2 \quad - f/c)/2 \tag{4.2}$$

The number of total distance computation is:

$$c \times s^2 \times (f^2/c^2 \quad + f/c)/2. \tag{4.3}$$

For any c >1:

$$s^2 \times (f^2 \quad - f)/2 \geq c \times s^2 \times (f^2/c^2 \quad - f/c)/2 \tag{4.4}$$

After the necessary simplifications:

$$f^2 > f^2/c \tag{4.5}$$

In the statement (4.4), the non-splitting approach (*FDBSCAN*) and the splitting approach (*M-FDBSCAN*) are compared in the context of number of distance computations. In Equation 4.5, it is shown that dataset splitting reduces the number of distance computations. Thus, even for sequential execution, the execution time performance of

our proposed algorithm is better than that of *FDBSCAN*.

After processing the c sub-datasets, the generated local sets of clusters are merged via the proposed merge procedure to establish the final set of clusters.

4.1.2. The Algorithm

Before the explanation of the algorithm, we want to mention about two words which are important and densely used. These are "*ε-neighborhood*" and "*density-reachable*". Two fuzzy data objects are in *ε-neighborhood* if the distance between them is less than or equal to ε. A pair of fuzzy data objects is density-reachable if there is at least one path between them, such that the distance between each of the adjacent nodes is in the ε-neighborhood. To assess the density-reachability of a pair, *sample matrices* for the relavant fuzzy data objects are constructed and the computation method of *reachability probability* described in Section 3.3.2, is utilized.

M-FDBSCAN is a "*divide and conquer*" algorithm and basically composed of three main steps:

- Dataset splitting
- Concurrent clustering
- Merging sub data regions

4.1.2.1. Dataset splitting. We gain significant performance improvement by splitting the original dataset into sub-datasets. While the number of distance computations is reduced by splitting method, as a side effect, the number of merge operations is increased. Since merge operation is computationally expensive, determining the optimum split number is very important. For several split numbers, execution times of the merge and the split operations are given in the experimental results section.

In a 2-D space, splitting a dataset is done with respect to certain horizontal

or vertical splitting lines. Sub-datasets are defiend according to the occurred regions between these lines. During the experiments, we test 2 different splitting approaches:

(i) *By a splitting algorithm:* For splitting, we develop a special algorithm derived from a binary search algorithm. The goal is to determine $n - 1$ lines that split the dataset into n nearly equal-sized sub-datasets. The execution time of the algorithm changes according to the distribution of the data in 2-dimensional space which means that distribution of the data determines the number of iterations required for getting the most appropriate splitting lines.

(ii) *By setting $n - 1$ equal-distance splitting lines for n sub-datasets:* The other approach is simply determining $n - 1$ equal-distance horizontal or vertical splitting lines for n sub-datasets.

In the experiments, we observe the execution times of both of the splitting approaches. In the case of normal data distribution, execution time of the second splitting approach is almost 0. But when the data is not well distributed, differences between the size of the sub datasets affect the total execution time performance negatively. When the dataset is split according to the first approach, even if a negligible amount of time is required for splitting,since the sub-dataset sizes are nearly the same, the overall execution time performance is better.

4.1.2.2. Concurrent clustering. After getting c sub datasets by one of the splitting methods, where c denotes the number of cores, each sub dataset is processed for clustering issues, by the assigned core concurrently. Since, some members of a cluster may be sited in another sub dataset during splitting the dataset, the generated clusters of the sub datasets are not at the final state. If the chosen c value is less than the optimum number, then the sub-datasets are apportioned among the cores (i.e. some of the cores processes more than one sub datasets for clustering issues.). As we prove, splitting improves the overall execution time performance, even for a single-core system.

4.1.2.3. Merging sub data regions. In the final step of the algorithm, the sub dataset regions are merged in order to get the final clusters. During the merge operations, the two essential constraints of density-based clustering approach are taken into consideration. These are:, i) each cluster must have at least μ cluster members and ii) among the members in a cluster, the distance can not be far than ε. In the algorithm, parallel lines in ε distance to the splitting line, are defined as ε-neighborhood lines. Each sub dataset has an ε-neighborhood line. The fuzzy data objects between an ε-neighborhood line and a splitting line are called as *insider fdos*. Instead of all of the fuzzy data objects of the sub datasets, traversing only the *insider fdos* and bringing the relevant density-reachable fuzzy data objects of *insider fdos* are sufficient for merging the sub dataset regions. This feature of the algorithm brings in a serious performans improvement.

The pseudocodes of all of the procedures of the algorithm are given formally in Section 4.1.3. But, since during the merge operations, it is required to handle many complex conditional clauses, for the sake of understandability, a comprehensive and semi-formal pseudocode of the merge procedure is also given here.

In Figures 4.4-4.7, f1 and f2 represent the *insider fdos* and C_x and C_y represent the clusters of the sub datasets D1 and D2, respectively. In Figure4.4 f1 is in C_x and f2 is in C_y and f1 is in ε-neighborhood of f2. In this case, C_x and C_y are merged under a new cluster C_{new} . In 4.5 and 4.6, again, f1 is in ε-neighborhood of f2 and one of the *insider fdos* is outlier and the other is in a certain cluster. In this case, the outlier insider fdo and the fuzzy data objects which are density-reachable from that outlier insider fdo are inserted into the other insider fdo's cluster. In Figure 4.7, the *insider fdos* f1 and f2 are both outlier. In this case, if the number of fuzzy data objects, which are density-reachable both from f1 and f2, is equal to or bigger than $(\mu - 2)$, then all these fuzzy data objects are inserted into a newly created cluster Cnew.

Merge operations can be done concurrently by assigning the adjacent sub dataset pairs to the available cores. The number of *insider fdos* determines the time complexity of the merge procedure.

```
Merge Sub-Data Regions
D1 , D2 : Sub dataset regions that will be merged
Cx : A cluster of D1
f1 : A fuzzy data object (of Cx) of D1
Cy : A cluster of D2
f2 : A fuzzy data object (of Cy) of D2
ε : Minimum distance between any pair of data objects in a cluster.
μ : Minimum number of fuzzy data objects in a cluster.
for all insider f1 of D1 do
    for all insider f2 of D2 do
        if f1 is in ε-neighborhood of f2 then
            if (f1 is in Cx) and (f2 is outlier) then
                Cx = Cx + (f2 + fdos of D2 density-reachable from f2)
            else if (f1 is outlier) and (f2 is in Cy) then
                Cy = Cy+ (f1 + fdos of D1 density-reachable from f1)
            else if (f1 is in Cx) and (f2 is in Cy) then
                Cnew =Cx + Cy
                Dispose Cx, and Cy
            else if (f1 is an outlier) and (f2 is an outlier) then
                OutlierSum1=|fdos of D1 density-reachable from f1| + 1
                OutlierSum2=|fdos of D2 density-reachable from f2|+ 1
                if (OutlierSum1 + OutlierSum2) ⩾ μ then
                    Create a new cluster Cnew
                    Insert (fdos of D1 density-reachable from f1) into Cnew
                    Insert (fdos of D2 density-reachable from f2) into Cnew
                    Insert (f1) into Cnew, and (f2) into Cnew
                end
        end
    end
end
```

Figure 4.3. Merging sub-data regions algorithm.

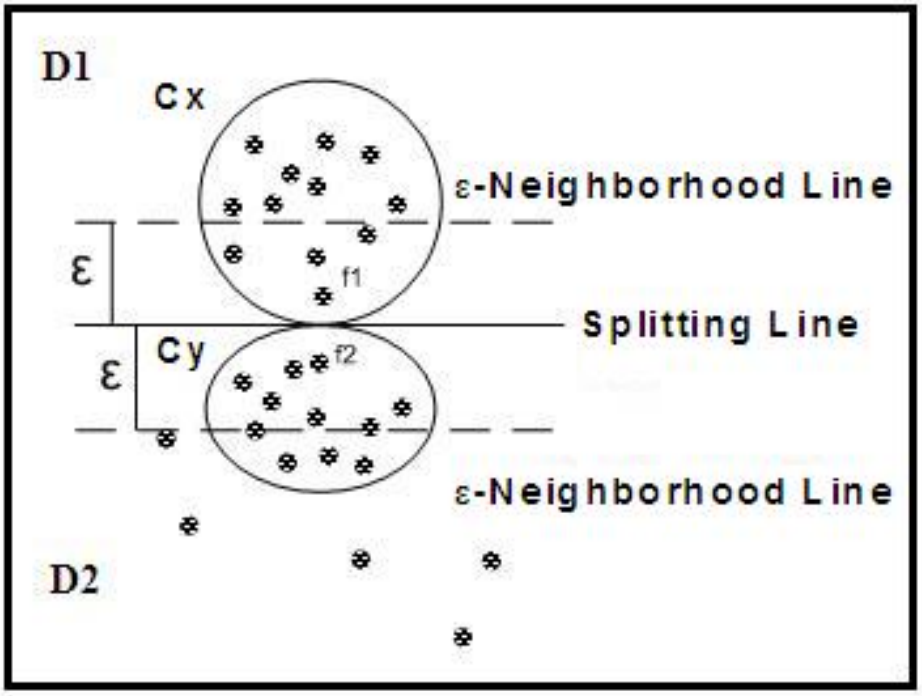

Figure 4.4. Case 1: f1 and f2 belong to different clusters.

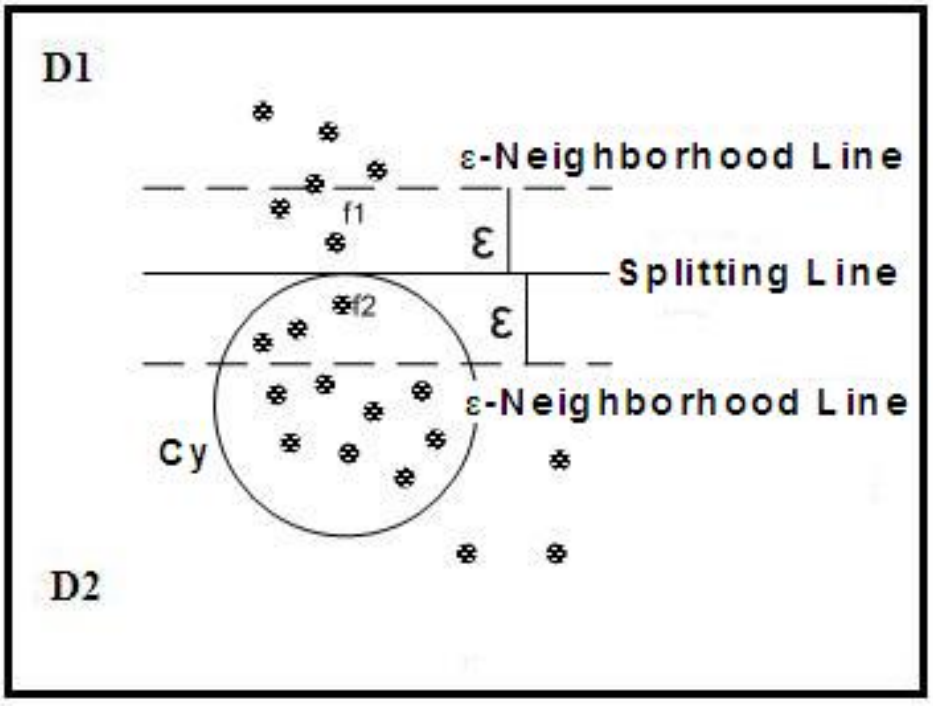

Figure 4.5. Case 2: f1 is outliers and f2 belongs to a cluster.

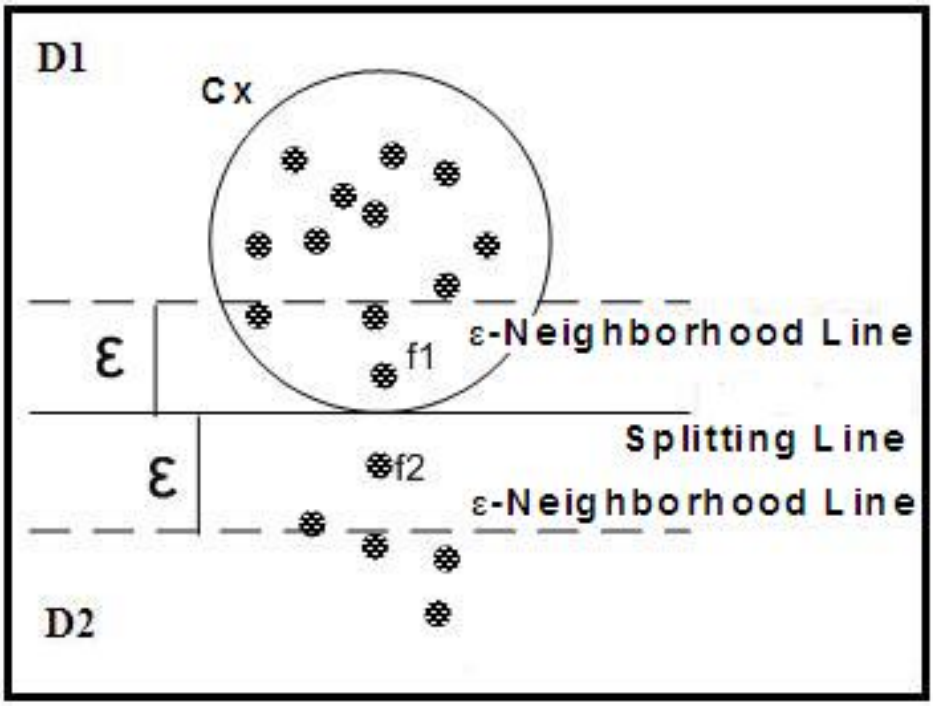

Figure 4.6. Case 3: f1 belongs a cluster and f2 is outliers.

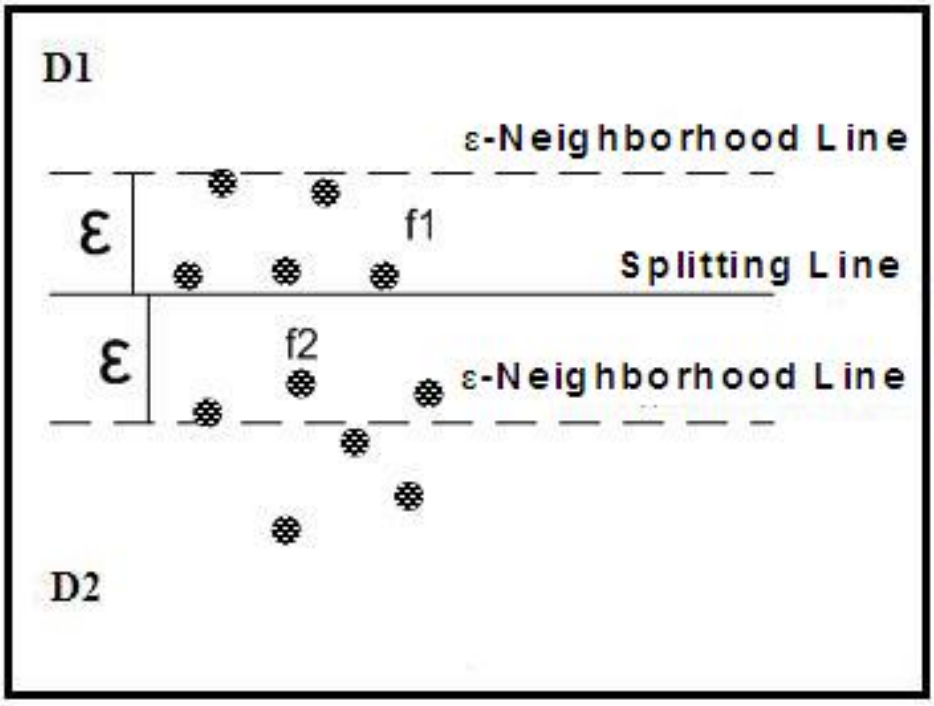

Figure 4.7. Case 4: f1 and f2 are both outliers.

4.1.3. Pseudocodes

In Figure 4.8, the original dataset D is split into two sub-datasets, D1 and D2, with a horizontal line. Since each fuzzy data object is represented by a sequence of sample points, a splitting line may also split a fuzzy data object. Thus, the sample points may be scatterred into two different sub-datasets. In this case, to avoid ambiguity, the fuzzy data object with all its sample points is assigned to only one of the sub-datasets.

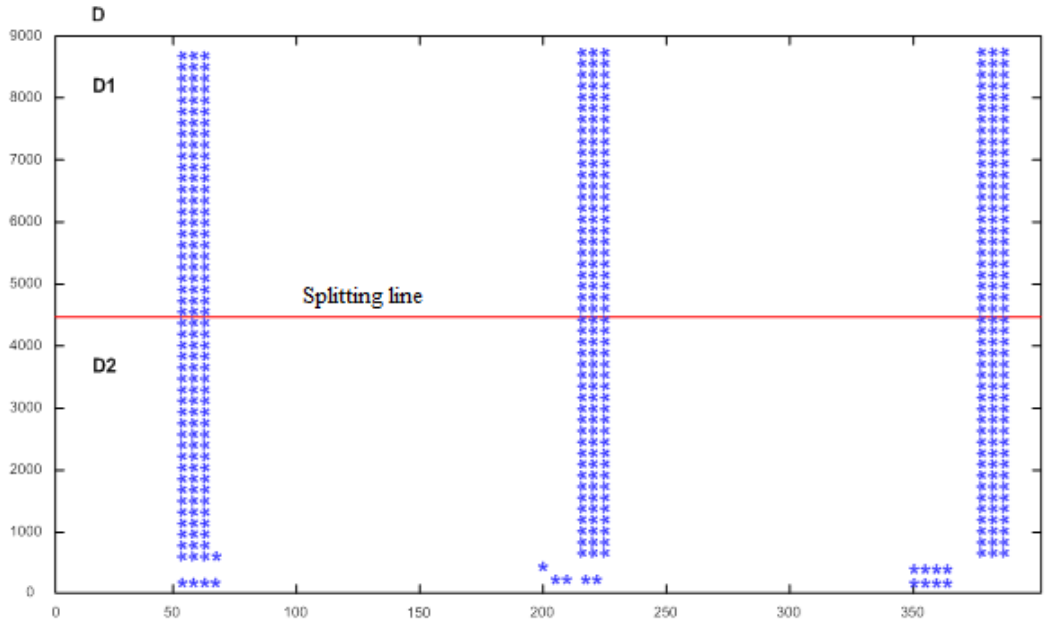

Figure 4.8. An illustration of dataset splitting.

```
input :
        D: Dataset of 2-dimensional fuzzy data objects.
        ε : Minimum distance between any pair of objects in a cluster.
        μ : Minimum number of fuzzy data objects in a cluster.
        c: Number of cores.
        D[i]: i^th sub-dataset.
M-FDBSCAN(D , ε , μ, c);
SplitDataset(c);
// Beginning of the parallel section
for i ← 1 to c do
    FDBSCAN(D[i] ,ε ,μ);
end
// End of the parallel section
// k :  index of core starting from 1
mergedDatasets ← D[1];
for j ← 1 to log2c do
    // Beginning of the parallel section
    for k ← 1 to c do
        if  k mod 2^j = 0 then
            MergeDatasets(D[k], D[k-2^(j-1)],ε,μ) ;
        end
    end
    // End of the parallel section
end
```

Figure 4.9. *M-FDBSCAN* algorithm.

```
1  Split_Dataset(c) // Split_Dataset is an implementation of a binary
       search like splitting algorithm
2  Sub-dataset_Size := Size(D)/c; // Approximate size of dataset
3  m[1]: = min_y; // min_y is the minimum y value in dataset D
4  for i ← 2 to c do
5      min_temp := m[i-1] ;
       // max_y is the maximum y value in the original dataset D
6      max_temp := max_y ;
7      Temp_Size := 0 ;
8      while ((Temp_Size <(Sub-dataset_Size – delta) or Temp_Size
       >(Sub-dataset_Size) + delta)) and (min_temp ⩽ max_temp) do
           // delta is an approximation parameter for decreasing the
              number of search steps
9          mid := min_temp + (max_temp – min_temp) / 2;
10         y1 := m[i – 1] ;
11         y2 := mid ;
12         Temp_Size := Number of fuzzy data objects in the region that is between
           y1 and y2 lines;
13         if Temp_Size <Sub-dataset_Size then
14             min_temp := mid +1;
15         else
16             max_temp := mid - 1;
17         end
18     end
19     D[i-1] ← The region between y1 and y2lines ;
20     m[i]: = mid ;
21 end
22 y1 := mid ;
23 y2 := max_y ;
24 D[c] ← The region between y1 and y2 lines;
```

Figure 4.10. *Split Dataset* procedure.

```
1  Merge_DataSets(D1, D2, ε, μ) // Merges 2 datasets by means of
       cluster regions and outlier fuzzy data objects.  C(f_i) is the label
       of the cluster region of fuzzy data object fi in Di, If C(f_x) is a
       null value, then this means that f_x is a noisy data object.
2  Select fuzzy data objects into R1 from D1 where y ⩾ max(y)–ε;
3  foreach non-visited fuzzy data object f1 in R1 do
4      foreach non-visited fuzzy data object f2 in R2 do
5          if fuzzy_distance(f1,f2) ⩽ ε then
6              if C(f1) is not null and C(f2) is not null then
7                  Change the cluster label of all fuzzy data objects of which the cluster
                   label is C(f1) in D1 as C(f2);
8              else if C(f1) is null and C(f2) is not null then
9                  Set the cluster label of f1 as C(f2);
10                 Set also the cluster label of all outlier fuzzy data objects that are
                   density-reachable from f1 as C(f2).
11             else if C(f1) is not null and C(f2) is null then
12                 Set the cluster label of f2 as C(f1);
13                 Set also the cluster label of all outlier fuzzy data objects that are
                   density-reachable from f2 as C(f1)
14             else if C(f1) is null and C(f2) is null then
15                 if (∑(density reachable data objects from f1) + 1 + ∑(density
                   reachable data objects from f2) + 1) ⩾ μ then
16                     Create a new cluster label C_new and set the cluster label of f1,
                       f2, all fuzzy data objects that are reachable from f1, and all fuzzy
                       data objects that are reachable from f2 as C_new
17             end
18         end
19     end
20 end
```

Figure 4.11. *Merge Dataset* procedure.

4.1.4. Experimental study

In our experiments, we use synthetic datasets of sizes 1000, 5000, 10000, 25000, and 50000. These datasets include 2-dimensional fuzzy data objects. Each fuzzy data object is represented by 7 sample points which are generated by using Gaussian probability function. The tests are done on an Intel Xeon X5650 2.67 GHz 24 Core CPU and 72 GB RAM server. The operating system of the computer is a 64-bit Linux Centos 5.5. The performances of *FDBSCAN* and *M-FDBSCAN* algorithms are compared with the predefined ε and μ parameter values of 120 and 7, respectively.

According to the results of the tests, even for single core systems, of splitting approach, the execution time performance is improved linearly, through splitting method. The tests are done for 2, 4, 8, 16, and 24 split sub-datasets.

As it is given in Table 4.1, the execution time performance of *M-FDBSCAN* and *FDBSCAN* are observed and compared according to several parameters. These are:

- Number of cores
- Number of splits
- Dataset size

Table 4.1. Observation results.

Cores	Splits	Dataset size	M-FDBSCAN			FDBSCAN (s)
			Merge (s)	With a split algorithm (s)	Without a split algorithm (s)	
1	2	1,000	0.0001	0.7033	0.7037	1.3809
1	2	5,000	0.0027	10.8570	10.8597	20.1277
1	2	10,000	0.0123	38.8254	38.8310	78.1639
1	2	25,000	0.0712	235.8166	235.8287	474.1698
1	2	50,000	0.2587	938.8371	938.8702	1884.3100
1	4	1,000	0.0007	0.3709	0.3713	1.3809
1	4	5,000	0.0086	6.0793	6.0820	20.1277
1	4	10,000	0.0369	19.2415	19.2453	78.1639
1	4	25,000	0.2175	119.0394	119.0551	474.1698
1	4	50,000	0.7744	470.4619	470.4952	1884.3100
1	8	1,000	0.0017	0.1847	0.1851	1.3809
1	8	5,000	0.0201	3.4781	3.4807	20.1277
1	8	10,000	0.0708	11.0671	11.0728	78.1639
1	8	25,000	0.4590	59.8739	59.8897	474.1698
1	8	50,000	1.8695	239.8696	239.9029	1884.3100
1	16	1,000	0.0040	0.0983	0.0988	1.3809
1	16	5,000	0.0631	2.2030	2.2056	20.1277
1	16	10,000	0.1671	6.1734	6.1790	78.1639
1	16	25,000	1.0077	31.0722	31.0828	474.1698
1	16	50,000	4.0320	124.0828	124.1158	1884.3100
1	24	1,000	0.0058	0.0698	0.0703	1.3809
1	24	5,000	0.1243	1.6332	1.6358	20.1277
1	24	10,000	0.2548	4.2852	4.2901	78.1639
1	24	25,000	1.5837	22.5020	22.5179	474.1698
1	24	50,000	6.2287	87.0127	87.0462	1884.3100

Continued on next page

Table 4.1. – Observation results (cont.).

Cores	Splits	Dataset size	M-FDBSCAN			FDBSCAN (s)
			Merge (s)	With a split algorithm (s)	Without a split algorithm (s)	
2	2	1,000	0.0001	0.3760	0.3765	1.3809
2	2	5,000	0.0029	6.1460	6.1487	20.1277
2	2	10,000	0.0133	20.1651	20.1707	78.1639
2	2	25,000	0.0667	127.9597	127.9703	474.1698
2	2	50,000	0.0989	305.3224	305.3428	1884.3100
4	4	1,000	0.0003	0.0954	0.0959	1.3809
4	4	5,000	0.0048	2.1324	2.1350	20.1277
4	4	10,000	0.0143	5.8992	5.9048	78.1639
4	4	25,000	0.0850	30.9325	30.9461	474.1698
4	4	50,000	0.2637	121.5938	121.6227	1884.3100
8	8	1,000	0.0004	0.0247	0.0251	1.3809
8	8	5,000	0.0053	0.5829	0.5855	20.1277
8	8	10,000	0.0168	2.1276	2.1333	78.1639
8	8	25,000	0.0777	8.7814	8.7973	474.1698
8	8	50,000	0.2998	32.1378	32.1629	1884.3100
16	16	1,000	0.0007	0.0141	0.0146	1.3809
16	16	5,000	0.0100	0.2602	0.2628	20.1277
16	16	10,000	0.0397	1.0904	1.0963	78.1639
16	16	25,000	0.1500	4.1285	4.1443	474.1698
16	16	50,000	0.5770	14.3250	14.3582	1884.3100
24	24	1,000	0.0287	0.0658	0.0663	1.3809
24	24	5,000	0.0253	0.2575	0.2603	20.1277
24	24	10,000	0.0747	0.6651	0.6707	78.1639
24	24	25,000	0.1760	2.8012	2.8169	474.1698
24	24	50,000	0.7157	8.6503	8.6754	1884.3100

Continued on next page

Table 4.1. – Observation results (cont.).

Cores	Splits	Dataset size	M-FDBSCAN			FDBSCAN (s)
			Merge (s)	With a split algorithm (s)	Without a split algorithm (s)	
24	24	1,000	0.0287	0.0658	0.0663	1.3809
24	24	5,000	0.0253	0.2575	0.2603	20.1277
24	24	10,000	0.0747	0.6651	0.6707	78.1639
24	24	25,000	0.1760	2.8012	2.8169	474.1698
24	24	50,000	0.7157	8.6503	8.6754	1884.3100

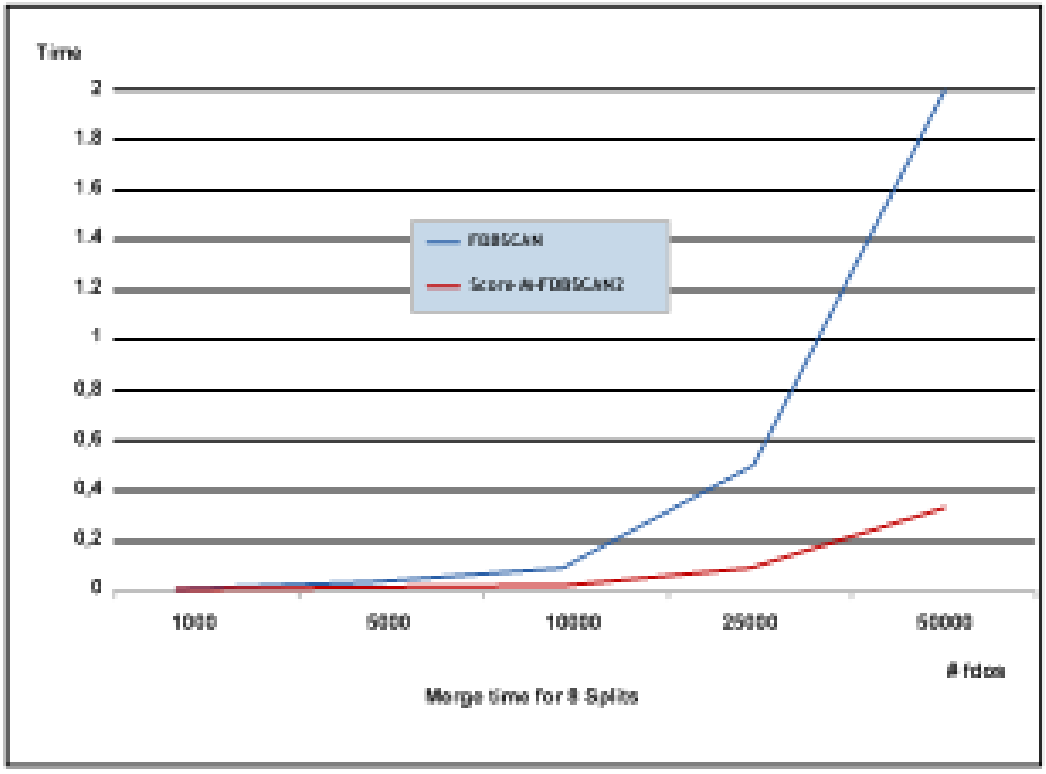

Figure 4.12. For 50,000 *fdo* records, the execution times of *FDBSCAN*, *M-FDBSCAN* (single-core), and *M-FDBSCAN* (multi-core).

We observe that while the number of splits increasing,the total execution time is reduced significantly. We notice that determining the optimum split number is very

important and must be subject of another research study. We also test 2 splitting methods. These are simply splitting i) with an algorithm and ii) without an algorithm. We record a negligible amount of performance improvement when the splitting is done with a certain algorithm.

In Figure 4.12, execution time comparison of *M-FDBSCAN* and *FDBSCAN* is given, for the single and the multi core modes. For the single core mode, the parameter, "*number of cores*", represents the number of splits in *M-FDBSCAN*. As an observation result, we can say that with the strength of the splitting method, *M-FDBSCAN* is faster than *FDBSCAN*, even in the single core mode. Obviously, in the multi-core mode, *M-FDBSCAN* provides the best execution time performance. In 4.13, merging time performance of *M-FDBSCAN* for the single and the multi-core modes are given. In the single core mode, the merge operations are done sequentially, while in the multi-core mode, they are parallelized.

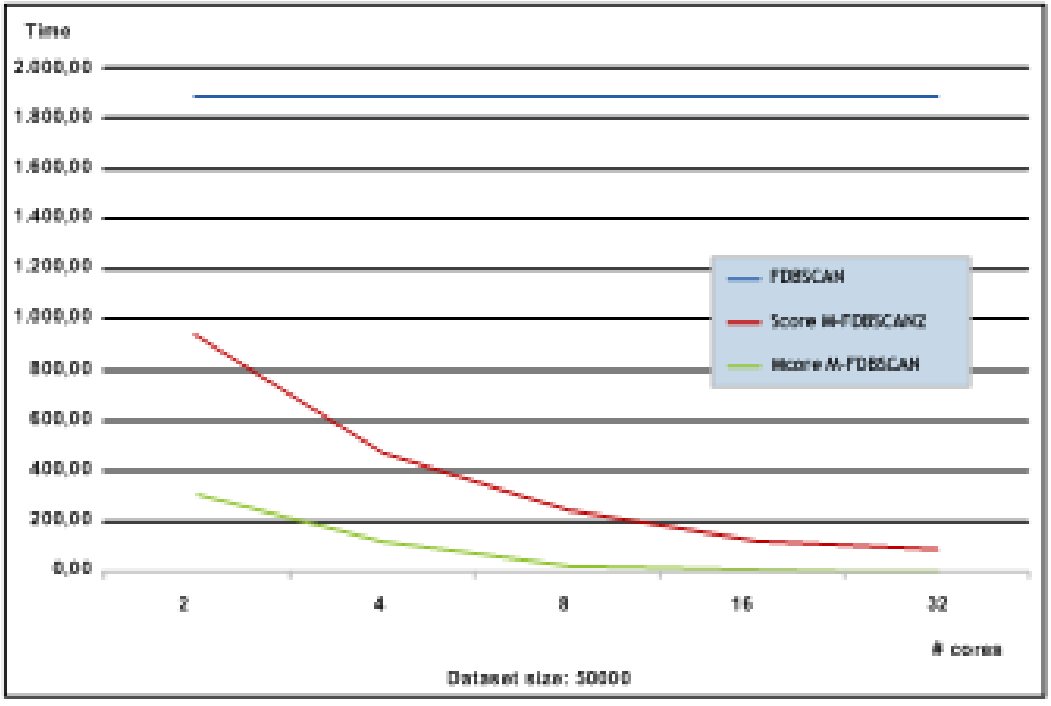

Figure 4.13. For 8 splits, the merging times needed for *M-FDBSCAN* for single-core and *M-FDBSCAN* for multi-core systems.

Several comments on the results of observations are:

(i) Linear performance improvements can be achieved by increasing the number of cores and splits.
(ii) The single core performance converges to a multi-core performance with the strength of the splitting method.
(iii) For the bigger number of splits, merge operation performance becomes critical for the total execution time performance. By parallelizing the merge procedure, the execution time is reduced considerably.
(iv) We observe that the performance gap between *M-FDBSCAN* and *FDBSCAN* enlarged quickly when the dataset size goes over 10,000.

4.2. An Evolutionary Clustering and Prediction Model for Gene Expression Data

As it is mentioned in Section 2.2, DNA microarray experiments are frequently utilized in the literature due to their various advantages. However, gene expression data obtained from DNA microarray experiments is noisy and consequently the computations based on such noisy data may lack accuracy. In this part of the thesis study, we propose an evolutionary clustering algorithm, *E-MFDBSCAN* and a prediction model. Our proposed methodology can be successfully applied on noisy gene expression data. In our methodology, global patterns of time series data can be extracted by the help of our evolutionary clustering approach. These patterns are used in predicting the future similarities of the global gene expression clusters. Thus, in this study, we propose a prediction model by constructing an autoregressive time series function (using these patterns).In the below sections, firstly we give an elaborated information about *E-MFDBSCAN* algorithm and then explain the proposed prediction model. At the end of the section, the relevant tests done for observing the performance of the proposed methodology and their results are presented.

4.2.1. E-MFDBSCAN

DNA microarrays are widely used in measuring expression levels of large number of genes. The strength of microarray experiments is due to their ability to accomplish many genetic tests in parallel. Due to the biological variations and the nature of probe-level measurement, missing and noisy data is a major problem in microarray experiments. Several studies are focused on handling the missing data problem. In Troyanskaya *et al.* [39], three missing data estimation methods: K-nearest neighbors (KNN), Singular Value Decomposition (SVD) and row average are tested and evaluated using three different gene expression datasets. In order to predict the missing gene expression time-series data, linear interpolation is proposed by Aach and Church [40]. Similarly, D'haeseleer *et al.* [41] propose a spline interpolation method for the estimation of missing time points. Bar-Joseph *et al.* [42] model gene expression profiles as a cubic spline. Using spline curves, missing data estimation is done. However, since the measured data is fuzzy, even if the missing data is estimated somehow, it must still be assumed as uncertain. Sivriver *et al.* [43] propose a dynamic modeling approach, DynaMiteC (Dynamic Modeling and Clustering), to overcome both noisy and missing data problems. According to their approach, time-course gene expression profiles are modeled and clustered using biologically meaningful parameters (e.g. point of induction). With the same motivation, another modeling approach, which is an extension of Gaussian Mixture model and named as PUMA-CLUST (Propagating Uncertainty in Microarray Analysis-CLUSTering) is proposed by Liu [44]. Because of using manual methods in some parts of gene expression experiments, reliability of the data is low. If this data is directly utilized as input to a data mining algorithm or a model for evaluating gene expression data, then the adverse affects to the desired results will be inevitable. In order to eliminate aforementioned adverse affects and reduce the fuzziness, we represent the gene expression data with sequence of sample points that are generated using uncertain data management techniques. By modeling uncertainty, our approach focuses on handling the noisy data problem rather than that of missing data. Since *E-MFDBSCAN* is a density-based clustering algorithm, it returns more accurate results for noisy datasets [6], compared to the other algorithms that are based on different approaches.

Gene expressions change especially during the transcription and the translation stages. If clustering is done for only static time points, the generated clusters help to deduce only instantaneous and local information about the overall cellular events. For vigorously understanding and analyzing the global patterns of gene activities, besides handling the uncertainty problem of gene expression data, a time series based dynamic data processing should be employed to be able to extract relevant information. Thus, recent studies are focused on evolutionary clustering and time-based dynamic modeling.

Subhani *et al.* [45] propose an evolutionary clustering algorithm, *EMMA*, that combines the approaches of Expectation Maximization (EM) and multiple alignment of gene expression profiles to cluster microarray time-series data. The algorithm uses k-means clustering algorithm, which is a centroid-based clustering algorithm. But the main drawback of the centroid-based representation is that the number of clusters has to be specified in advance. Because of the nature of the data domain, determining the number of clusters before the clustering process is difficult. Another evolutionary clustering algorithm, *EvoCluster* is proposed by Ma *et al.* [46]. Proposing a fitness function, *EvoCluster* aims to extract the global patterns of gene expression data. This method is also effective for handling noisy data. However, the fitness function is not a time-series function. This means that the time parameter is not considered during clustering. Thus, the generated global clusters are not time-based.

Holter and Maritan [47] propose a dynamic model for gene expression data. Singular Value Decomposition (SVD) approach is used in their model which is a good idea to define the characteristic modes that capture the essential time-based features of the gene expression profiles. The obtained average expression patterns of each cluster are used to construct the best fit time translational matrices. However, because of the generality of this approach, it is possible to ignore the valuable inter and intra cluster relationships. Zhao *et al* [48] propose statistical models to extract the genes regulated by the cell cycle. For each periodically analyzed specific dataset, a custom model is generated. This is the distinct weakness of their proposed method. Aach and Church [40] focus on aligning gene expression time series. They propose time warping methods against clustering. Global pattern extraction and prediction are not possible

with these algorithms.

The proposed algorithm, *E-MFDBSCAN*, is an evolutionary clustering algorithm which is based on the clustering method of DBSCAN. The uncertain data clustering method proposed in *FDBSCAN* and the parallel computation method proposed in our *M-FDBSCAN* algorithm are used in *E-MFDBSCAN*. As it is mentioned in the previous sections, the clustering approach proposed in *FDBSCAN* is based on the *sample matrix* concept of which extended version, named *t-sample matrix*, is also proposed and used in *E-MFDBSCAN*.

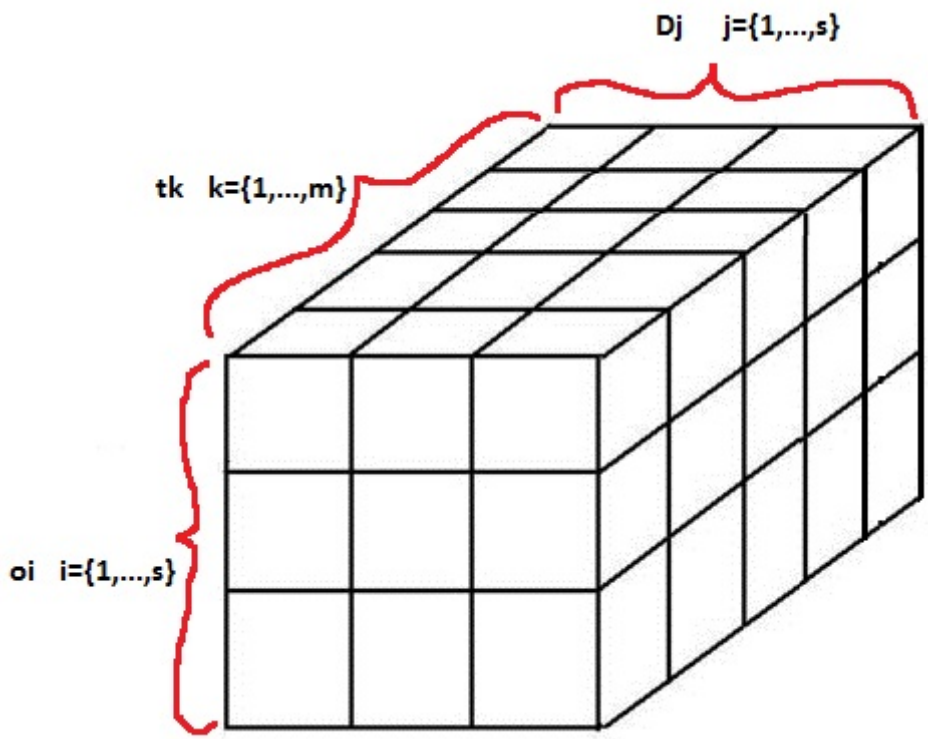

Figure 4.14. A *t-sample matrix* with $m = 5$ time points and $s = 3$ sample data points.

Since our evolutionary clustering model is developed for gene expression time series data, we extend the *sample matrix* concept by adding time dimension. The new matrix, *t-sample matrix*, is a three dimensional $m \times s \times s$ matrix, where m denotes the number of points of time dimension and s denotes the number of derived sample points for fuzzy data objects. In *E-MFDBSCAN*, for each fuzzy data

object x, a sequence of $m \times s$ samples, $<x_{11},\ldots,x_{m1},\ldots,x_{ms}>$is derived by using Gaussian probability density function. Similar to *sample matrix*, in *t-sample matrix*, o_{ki} represents the ith derived sample point for the kth time point, such that $\{o_{ki}|<o_{11},\ldots,o_{1s},\ldots,o_{ki},\ldots,o_{ms}> \in o\}$and also D_{kj} represents the jth database instance for the the kth time point, where $\{x_{kj}|< x_{11},\ldots,x_{1s},\ldots,x_{ki},\ldots,x_{ms} > \in D \wedge x_{kj} \neq o_{kj}\} \cup o_{ki}$. Finally, each matrix element is $m_{k,i,j} =|N_{\varepsilon,Dkj}(o_{ki})|$, where $N_{\varepsilon,Dkj}(o_{ki})$ denotes the set $\{x_{kj}/d(o_{ki},x_{kj})\leqslant \varepsilon \wedge x_{kj} \in D_{kj}\}$ and $d(.,.)$ denotes the chosen distance function. In Figure 4.14, a $5 \times 3 \times 3$ *t-sample matrix* is illustrated.

Core Object Probability

Begin

(i) *For each time point t_k, where $\{t_k/<t_1,\ldots,t_m>\}$do;*

  (a) *Decrease the $m_{k,i,j}$ by 1 for which $d(o_{ki},x_{kj})\leqslant \varepsilon$ holds.*

  (b) *Count the number of elements int the t-sample matrix T-SM(o) which contain values higher or equal to $\mu \quad -1$.*

  (c) *Normalize the result by s^2.*

  (d) *Assign the final normalized result to $P^{core}(o)_k$*

(ii) *Count the core probabilities which are higher than or equal to 0,5.*

(iii) *Normalize the result by m*

(iv) *Assign the final normalized result to $P^{core}(o)_{final}$.*

End

Figure 4.15. Procedure to compute *Core Object Probability.*

In the first step of the algorithm, the *t-sample matrices* are constructed for all of the fuzzy data objects. A *t-sample matrix* T-SM(o) where o is any fuzzy data object, is a data structure utilized to compute the *reachability probabilities* between o and the other fuzzy data objects in the database. In short, *reachability probability* w.r.t. o and x denotes the probability of the situation that o and x are in the same cluster. Assigning an object to a cluster or flagging it as an outliers are all decided according

to the value of the *reachability probability.*

$$P^{reach}(x, o) = P^{core}(o) \bullet P_d(x, o)(\varepsilon) \tag{4.6}$$

In the Equation 4.6, $P^{core}(o)$ represents the *core object probability* of o which denotes the probability of the situation that o has at least μ*-1* neighbors in ε-neighborhood, except x. This probability value is derived using *t-sample matrix.* We customize the derivation procedure as in algorithm given in Figure 4.15.

Finally, the next probability that must be computed is the *distance distribution probability* $P_d(x,o)(\varepsilon)$ w.r.t. x and o, which denotes the probability of the situation that x is in ε-neighborhood of o. As in the core object probability computation, the computation of $P_d(x,o)(\varepsilon)$ is customized for *E-MFDBSCAN,* given in algorithm given in Figure 4.16.

Distance Distribution Probability

Begin

(i) *For each time point t_k of the fuzzy data object x, where $\{t_k/<t_1,\ldots,t_m>\}$do;*

  (a) *Count the number of events $d(o_{ki},x_{kj}) \leqslant \varepsilon$.*

  (b) *Normalize the result by s^2.*

  (c) *Assign the final normalized result to $P_d(x,o)(\varepsilon)_k$*

(ii) *Count the distance distribution probabilities which are higher than or equal to 0,5.*

(iii) *Normalize the result by m*

(iv) *Assign the final normalized result to $P_d(x,o)_{final}$.*

End

Figure 4.16. Procedure to compute *Distance Distribution Probability.*

Thus, the final reachability probability is

$$P^{reach}(x,o)_{final} = P^{core}(o)_{final} \bullet P_d(x,o)_{final}. \tag{4.7}$$

If $P^{reach}(x,o)_{final} \geqslant 0{,}5$ then it is accepted that x is reachable from o. This also means that x and o are in the same cluster. If for all x, $P^{reach}(x,o)_{final} < 0{,}5$, then o is flagged as outliers With this new perspective proposed by *E-MFDBSCAN*, time-based global clusters and outliers can be generated.

Since, the standard distance functions, like *Euclidean*, which are mostly used for the objects in n-D spaces (n $\geqslant$ 2), are not adequate for defining the correlations among gene expression data which is in 1-D space. Instead of them, *Pearson* correlation function is used in many gene expression researches [39, 49–51]. Thus we also choose *Pearson* correlation as distance measure.

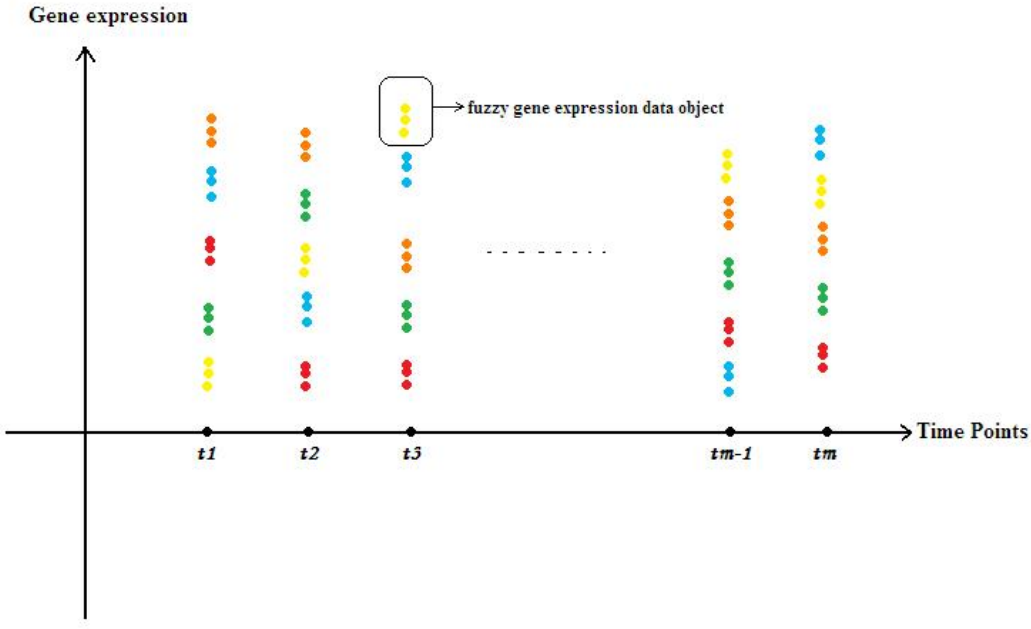

Figure 4.17. An illustration of gene expression time series database.

In Figure 4.17, an illustration of a gene expression time series database is given. As it is seen, for each time point, the gene expression data lies on a vertical line and each fuzzy data object is represented with a sequence of derived sample points.

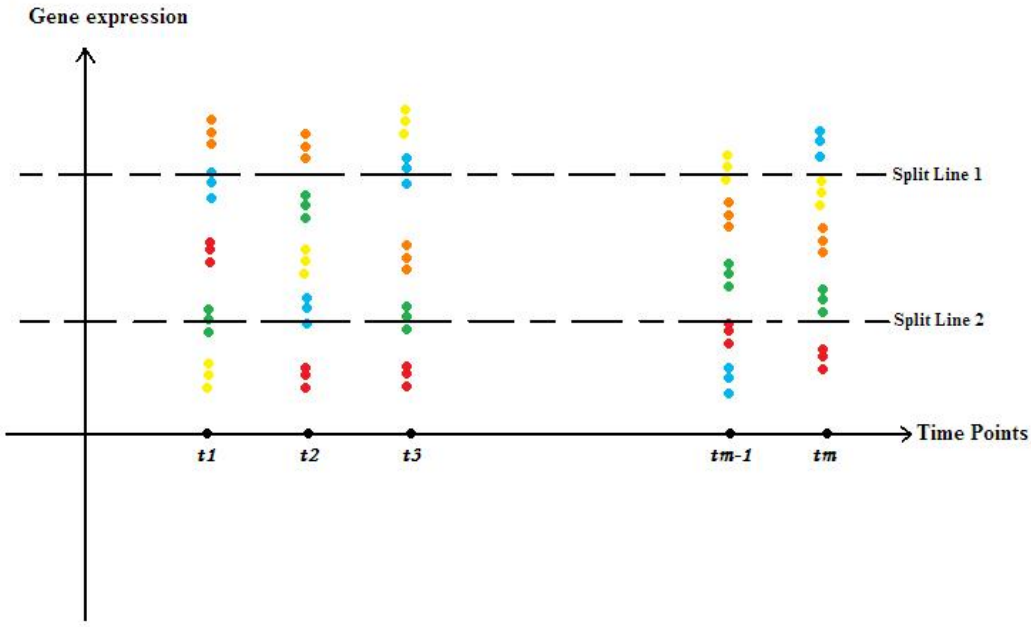

Figure 4.18. Gene expression time series database with *splitting lines*.

4.2.1.1. The Algorithm. *E-MFDBSCAN* algorithm is an advanced and composite variant of *FDBSCAN* and *M-FDBSCAN* algorithms of which the goal is to generate time-based evolutionary sets of clusters. The sample matrix construction and the reachability probability computation procedures are inherited from *FDBSCAN* and the database splitting (as illustrated in Figure 4.18) procedure, and the merge procedure which is applied for the temporary sets of clusters are inherited from *M-FDBSCAN*. All the inherited procedures are customized according to our proposed evolutionary clustering concept.

If *E-MFDBSCAN* is assumed as a function, then it returns a set of sets of clusters SC, such that; SC=$\{SC_1, \ldots, SC_i, \ldots, SC_m\}$where m denotes the number of time points and SC_i denotes the set of clusters generated at the time point t_i using the database Di $\subseteq$ D, where Di denotes the database in which the data for the time points $<t_1, \ldots, t_i>$are stored.

```
input :
D: Database, m: Number of time points, n: Number of sub-databases for
parallel execution, μ : Maximum number of objects in a cluster, ε: Minimum
distance between two objects in a cluster
E-MFDBSCAN (D, m, n, μ , ε, s) Returns SC
for i ← 1 to m do // Derives Di ⊆ D so that it includes the data
values of the time points of <t1,...,ti>
    Di := Derive_Di(D,i);
    SCi := {}
end
// Initializes the set of sets of clusters
SC := {}// Construction of the set of sets of clusters SC
for i ← 1 to m do // Splits Di into n optimum sub-databases
    Split_Database (Di, n) // Start of paralel execution
    for k ← 1 to n do // Constructs t-sample matrices for all fuzzy
    data object fdo in Dik
        Construct_t_sample_matrix(Dik) // Constructs temporary
            set of clusters SCik acc.  to reachability probabilities
        // P^reach(x,o), for each x and o where x and o denote fuzzy
            data objects in Dik and x≠o
        SCik := Construct_Temp_Clusters(Dik)
    end
    // End of the parallel section  // Merges the n temporary set of
    clusters <SCi1,...,SCin>to get the final set of clusters
    for k ← 1 to n − 1 do
        SCi := Merge_SCi(SCik , SCik+1)
    end
    SC := SC + SCi // Adds the constructed set of clusters SCi to
        the set of sets of clusters SC
    Return SC
end
```

Figure 4.19. *E-MFDBSCAN* procedure.

4.2.2. The Prediction Model

Recently, several studies are focused on gene expression profile prediction. The dynamic model proposed by Holter and Maritan [47] has a capability of class prediction. The class prediction model proposed by Sorlie *et al.* [52] is based on *prediction analysis of micro arrays* (PAM), which is a variant of nearest-centroid classification approach. Also another classification and prediction solution based on nearest-centroid approach was proposed by Tibshirani [53]. But, the common restriction of the models is that they are applicable if and only if the classes are known and defined. Unfortunately, in most of the cases, the classification of gene expression profiles is a challenging problem. As mentioned in [53], since the number of genes to be classified and predicted is much more than the number of samples achieved by micro array experiments, choosing the best-fit classes for each gene is nearly impossible. Also, significant identification of the genes that contribute to the classification is difficult. For these reasons, clustering-like unsupervised learning methods are widely used for grouping gene expression profiles.

In several studies, evolutionary clustering approach based prediction models are proposed. The evolutionary clustering approaches, in which the time parameter is not considered, like *EvoCluster* proposed by Ma et.al. [46], are not suitable for future-time pattern prediction models.

In our study, we propose a prediction model for gene expression time series. One of the crucial benefits of *E-MFDBSCAN* is to demonstrate temporal evolution of the gene expression clusters. Thus, this feature enables us to make prediction for a future time point. In Figure 4.20, time-based evolutionary changes of sets of clusters are illustrated.

To use in the proposed prediction model, we define a similarity measure between the two timely adjacent sets of clusters. The obtained similarity values determine the level of affinity of two sets of clusters. The proposed prediction model is represented with the time-series function which is constructed according to autoregressive (AR)

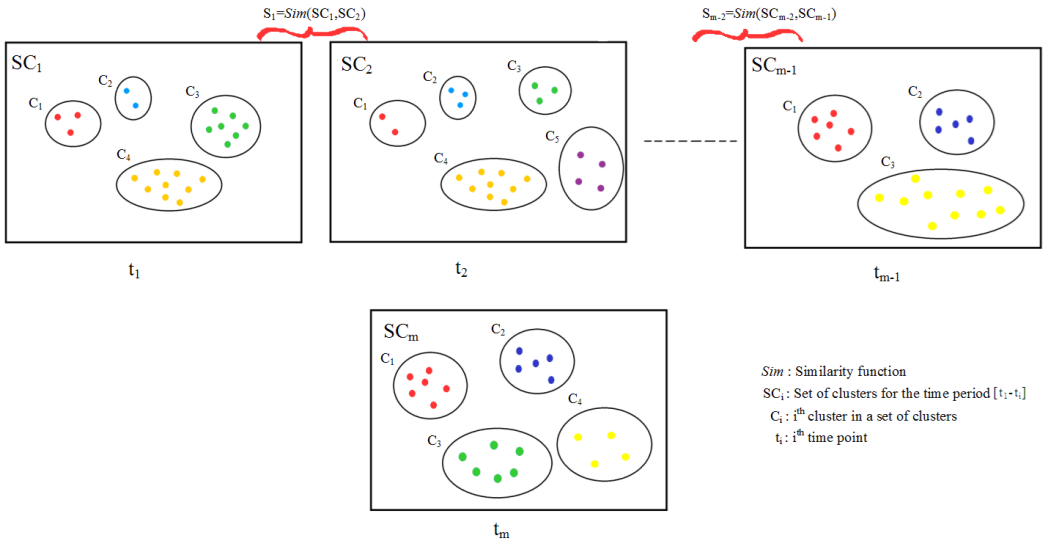

Figure 4.20. Illustration of time-based evolutionary change of sets of clusters.

model concept and the achieved time-based similarity values.

$$
\begin{aligned}
\Delta_i &= E - MFDBSCAN_i \\
\mathbf{Sim}(\Delta_i - \Delta_{i+1}) &= \mathbf{A} \\
&+ \mathbf{B} \times \mathbf{Sim}(\Delta_i - \Delta_{i+1}) \\
&+ \mathbf{E_{i+1}} \qquad i = 1, \ldots, m
\end{aligned} \tag{4.8}
$$

In Equation 4.8, "**Sim**" represents the *BestMatch* similarity function defined in Goldberg [54], which is utilized for computing the similarity values between two timely adjacent sets of clusters. "**A**" and "**B**" are the coefficients of the time series function and **E** is the noise function. The index i represents the ith time point. For observing the prediction performance of the model, the similarity value, for the next future time period, [t_m-t_{m+1}], is obtained from the time-series function. Then, the obtained similarity value is compared with the similarity value which is achieved by executing *E-MFDBSCAN* for the time points t_m and t_{m+1}. Notice that, though, t_{m+1} represents a next future time point, the gene expression data at this time point is known and reserved in the database for testing issues. The difference between these two similarity

values determines the performance of the prediction model.

4.2.2.1. BestMatch. As it is mentioned above, we use *BestMatch* for the computation of similarity values. It produces a similarity value between two sets of clusters. Let SC_1 and SC_2 be the sets of clusters, for each cluster $C_i \in SC_1$, the goal is to find the most similar pair, C' in SC_2 by using one to one correspondence. Symmetrically, also the most similar pair of each $C_j \in SC_2$ is searched for in SC_1. For this purpose, a difference function **d** was proposed, which returns the number of unmatched members, after one to one correspondence. The most similar pairs can be found by;

$$\mathbf{d}(C_i, C') = \min_{j=1,\ldots,m} \mathbf{d}(C_i, C'_j) \text{ where m denotes the number of members of } C'_j$$

and

$$\mathbf{d}(C_j, C') = \min_{i=1,\ldots,n} \mathbf{d}(C_j, C'_i) \text{ where n denotes the number of members of } C'_i$$

Thus, the total similarity value for the sets of clusters SC_1 and SC_2 is;

$$\mathbf{d}(SC_1, SC_2) = \sum_{i=1}^{n} \min_{j=1,\ldots,m} \mathbf{d}(C_i, C'_j) \; + \; \sum_{j=1}^{m} \min_{i=1,\ldots,n} \mathbf{d}(C_j, C'_i) \tag{4.9}$$

by normalizing this value with (m+n), we can get the final similarity value.

4.2.3. Results and discussion

Since time series of gene expression is commonly very short (i.e. 4 to 20 samples) and usually unevenly sampled [55], extracting the time-based evolutionary patterns is not so easy. In our study, two different time series gene expression data sets are used for testing the issues about evolutionary clustering and prediction.The evolutionary clustering tests are made for observing the patterns of the generated global clusters. For each time point i, the set of global clusters SC_i is established including the time

period [t_1-t_i]. Thus, for m time points, m sets of global clusters are established, such that $\{SC_i|<SC_1,\ldots,SC_i,\ldots,SC_m\}$. For the prediction, obtaining the similarity values of each two timely adjacent sets of global clusters, the coefficients and the noise of the time series function given in Equation 4.8 is defined. The performance tests of the prediction capability of the model are done due to the similarity values obtained from the time-series function. Then, these values are compared with the similarity values achieved by executing *E-MFDBSCAN* for two timely adjacent time points. The comparison results for the used datasets are given in Figures 4.21 and 4.22.

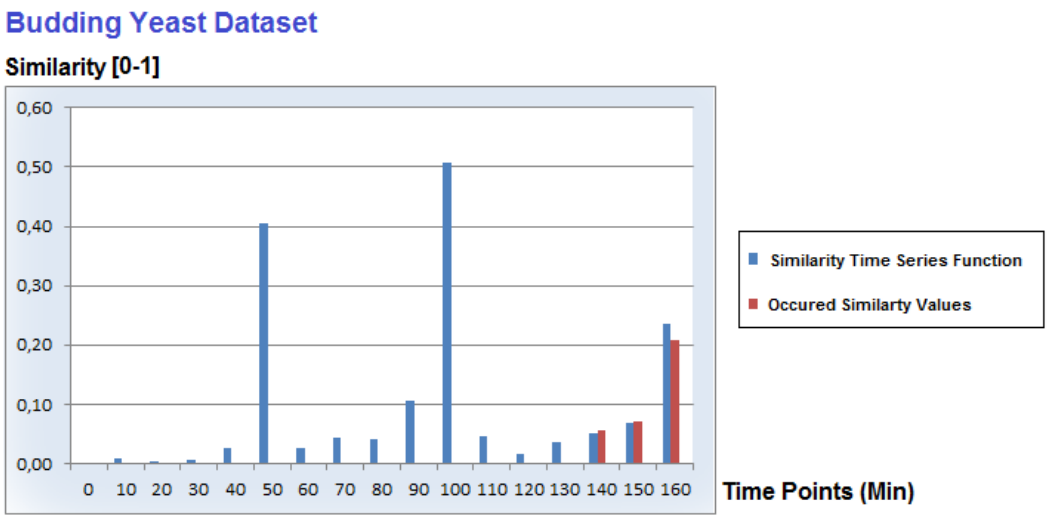

Figure 4.21. Similarities vs. Time graph for budding yeast dataset.

The first dataset is budding yeast dataset [56] that the gene expression time-series data is obtained during the yeast cell cycle. The dataset also includes the class information of the achieved data. For 6,220 yeast genes, expression levels are observed during the experiments [56]. At each 10 minutes, from the time point 0 to 160 minutes, totally 17 distinct temporal data is achieved for each gene. The used clustering algorithm is described in [39]. We extended this data set by generating 8 sample points for each given gene expression data. Since *E-MFDBSCAN* is a density-based clustering algorithm the input parameters ε (maximum distance between two members in a cluster) and μ (minimum number of members in a cluster) are derived from the given class information. The values of the parameters derived from the class information

are: $\varepsilon = 0.005$ and $\mu = 5$. Starting from the 0th time point, for each time point of 17 time points, *E-MFDBSCAN* is executed to generate the sets of global clusters. After computing the similarity values for each set of clusters pairs of adjacent time points, we get the set of similarity values, S. Using S, the unknown coefficients in Equation 4.8 are extracted. Finally, a time series function is established for the budding yeast dataset. During the tests, the similarity values for the last three time points are predicted. As it is shown in Figure 4.21, the actual and the predicted value bars are nearly at the same height. Due to the stable pattern of gene expression time series data, after the construction of a model with a reasonable number of time points, satisfactory prediction results can be achieved.

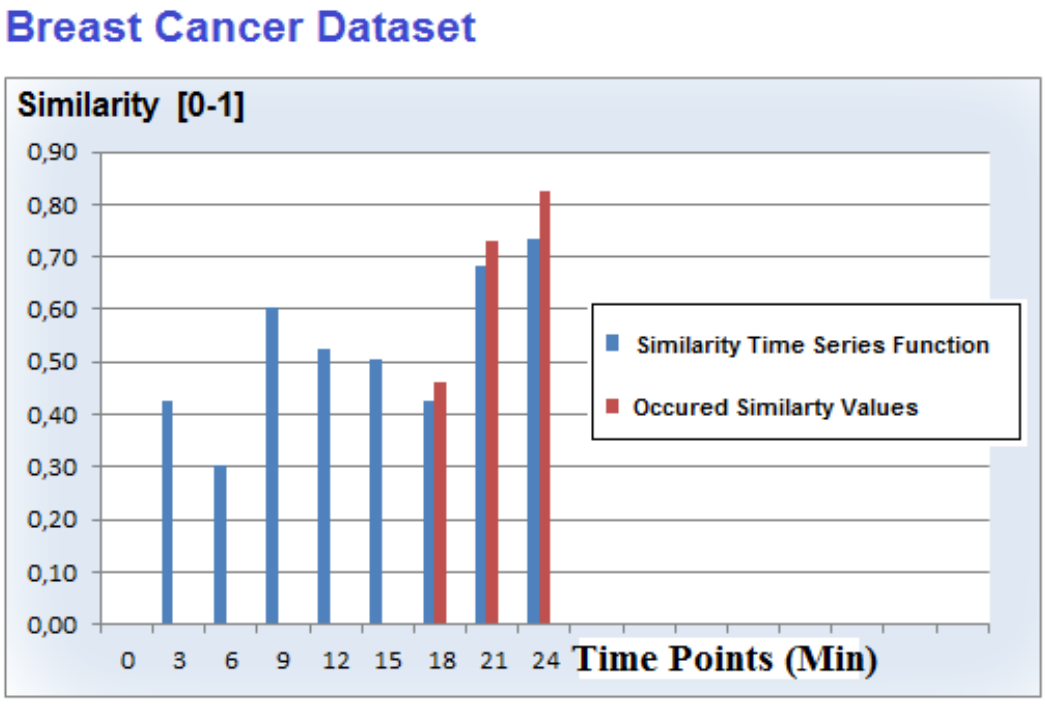

Figure 4.22. Similarities vs. Time graph for breast cancer dataset.

The next dataset was a breast cancer gene expression time series dataset [57, 58]. The data is the result of analysis of breast cancer MCF10A-Myc cells at four time points up to 24 hours following a treatment with dexamethasone to activate the glucocorticoid receptor (GR). The dataset is composed of 22283 gene expression time series values. For each gene expression data, again, 8 sample points are derived using Gaussian

probability density function. Contrary to the first dataset, the classes are not defined in this dataset. Thus, we do the tests with several μ and ε parameter values. The relevant graph is shown in Figure 4.22. As in the previous experiment, for the last three time points, the actual and the predicted values are compared. Since the number of time points is smaller than the first dataset, fitting ratio of the time series function is low. For this reason, the differences between the predicted and the occurred values are slightly bigger than the first dataset.

5. CONCLUSION

Due to the biological variations and the nature of probe-level measurement, missing and noisy data is a major problem in micro array experiments. Since gene expression data is achieved by micro array experiments, it can be classified as uncertain data. One of the uncertain data management methods is representing a fuzzy data with a certain number of sample points generated by a probability density function which we also utilize. In spite of the advantages, the method causes the original database to be enlarged with respect to the number of sample points. Thus, to speed up data processing, the relevant algorithms are needed to be adapted for multi-core systems. For this purpose, in the first phase of our study, we adapt a widely used uncertain data clustering algorithm for multi-core systems which is named *M-FDBSCAN*. Even though the algorithm is devised for multi-core systems, in case of single core executions, dramatic performance improvement was observed.

In the second phase, we propose a methodology to predict the evolutionary patterns of gene expression time-series data. For this purpose, we develop an evolutionary clustering algorithm, named *E-MDBSCAN*, and a prediction model. The distinct contribution of the methodology is the comprehensive solutions proposed for several problems of gene expression data processing. While some of the relevant problems are solved by utilizing the existent approaches and algorithms, for some others, novel algorithms and models are proposed. With this perspective, in literature, there are only a few studies that concentrate on a complete solution. Most of them focus on a specific problem of gene expression data processing.

In the clustering part of the methodology, we present an evolutionary clustering algorithm. The algorithm, named *E-MFDBSCAN,* generates time-based global gene expression clusters. The other features of the algorithm are (i) parallel execution capability and (ii) uncertain data consideration and management. According to the result of our literature survey, there is no other clustering algorithm that gathers all of the aforementioned features in a single algorithm.

In the prediction part of the methodology, we present a prediction model. The similarity values between two timely adjacent set of global clusters, generated by *E-MFDBSCAN*, are used to construct an autoregressive time-series function. The function is used for the prediction issues. Verification of the model is done by comparing the predicted and the actual similarity values for the same time period. The predicted similarity values are derived from the function and the actual similarity values are achieved by executing *E-MFDBSCAN* for two adjacent time points. It is observed that the prediction performance of the model depends on the size of the training dataset. Unfortunately, in the context of gene expression data, time series is commonly short. This means that the training dataset has only a few time points (e.g. 4 to 20). But, despite of this essential drawback, due to the stable pattern of gene expression time series data, the prediction results are satisfactory.

Although, the methodology is applied to gene expression data domain, it can be used for other data domains, such as social media data domain. As a future work, the model will be tested for social media analysis.

REFERENCES

1. Tamayo, P., D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander and T. R. Golub, "Interpreting Patterns of Gene Expression With Self-Organizing Maps: Methods and Application to Hematopoietic Differentiation", *Proceedings of the National Academy of Sciences*, Vol. 96, No. 6, pp. 2907–2912, 1999.

2. Kaufman, L. and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, Inc., Hoboken, New Jersey, 1990.

3. Kohonen, T., *Self-Organizing Maps*, Springer, Berlin, 1997.

4. Ester, M., H. Kriegel, J. Sander and X. Xiaowei, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise", *Proceedings of the Second International Conference Knowledge Discovery and Data Mining (KDD'96)*, pp. 226–231, ACM, 1996.

5. Fang, H. and J. Gough, "supraHex: An R/Bioconductor Package for Tabular Omics Data Analysis Using a Supra-Hexagonal Map", *Biochemical and Biophysical Research Communications*, Vol. 443, No. 1, pp. 285–289, 2014.

6. Mumtaz, K., Duraiswamy. and K., "An Analysis on Denstiy Based Clustering of Multi Dimensional Spatial Data", *Indian Journal of Computer Science and Engineering*, Vol. 1, No. 1, pp. 8–12, 2010.

7. Kriegel, H.-P. and M. Pfeifle, "Density-Based Clustering of Uncertain Data", *Proceedings of the Eleventh International Conference on Knowledge Discovery and Data Mining (KDD'05)*, pp. 672–677, ACM, 2005.

8. Erdem, A. and T. I. Gündem, "M-FDBSCAN: A Multi-Core Density-Based Uncertain Data Clustering Algorithm", *Turkish Journal of Electrical Engineering and*

Computer Sciences, pp. 143–154, 2014.

9. Weaver, R. H., *Molecular Biology*, McGraw Hill Education, Ohio, U.S.A, 2002.

10. Oliver, S., "A Network Approach to the Systematic Analysis of Yeast Gene Function", *Trends in Genetics*, Vol. 12, No. 7, pp. 241–242, 1996.

11. Roberts, R. J., "Identifying Protein Function–a Call for Community action.", *PLoS Biology*, Vol. 2, No. 3, pp. 293–294, 2004.

12. Altschul, S. F., T. L. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller and D. J. Lipman, "Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs", *Nucleic Acids Research*, Vol. 25, No. 17, pp. 3389–3402, 1997.

13. Shendure, J., R. D. Mitra, C. Varma and G. M. Church, "Advanced Sequencing Technologies: Methods and Goals", *Nature Reviews Genetics*, Vol. 5, No. 5, pp. 335–344, 2004.

14. Bork, P., T. Dandekar, Y. Diaz-Lazcoz, F. Eisenhaber, M. Huynen and Y. Yuan, "Predicting Function: From Genes to Genomes and Back", *Journal of Molecular Biology*, Vol. 283, No. 4, pp. 707–725, 1998.

15. Nguyen, D. V., A. B. Arpat, N. Wang and R. J. Caroll, "DNA Microarray Experiments: Biological and Technological Aspects", *Biometrics*, Vol. 58, No. 4, pp. 701–717, 2002.

16. Slonim, D. K., "From Patterns to Pathways: Gene Expression Data Analysis Comes of Age", *Nature Genetics*, Vol. 32, pp. 502–508, 2002.

17. Quackenbush, J., "Microarray Data Normalization and Transformation", *Nature Genetics*, Vol. 32, pp. 496–501, 2002.

18. Lee, I., S. V. Date, A. T. Adai and E. M. Marcotte, "A Probabilistic Functional

Network of Yeast Genes", *Science*, Vol. 306, No. 5701, pp. 1555–1558, 2004.

19. Bar-Joseph, Z., G. Gerber, D. K. Gifford, T. S. Jaakkola and I. Simon, "A New Approach to Analyzing Gene Expression Time Series Data", *Proceedings of the Sixth Annual International Conference on Computational Biology*, pp. 39–48, ACM, 2002.

20. Walker, M. G., W. Volkmuth, E. Sprinzak, D. Hodgson and T. Klingler, "Prediction of Gene Function by Genome-Scale Expression Analysis: Prostate Cancer-Associated Genes", *Genome Research*, Vol. 9, No. 12, pp. 1198–1203, 1999.

21. Abiteboul, S., P. C. Kanellakis and G. Grahne, "On the Representation and Querying of Sets of Possible Worlds", *Theoretical Computer Science*, Vol. 78, No. 1, pp. 158–187, 1991.

22. Kriegel, H.-P. and M. Pfeifle, "Hierarchical Density-Based Clustering of Uncertain Data", *Data Mining, Fifth IEEE International Conference on*, pp. 4–10, IEEE, 2005.

23. Ankerst, M., M. M. Breunig, H.-P. Kriegel and J. Sander, "Optics: Ordering Points to Identify the Clustering Structure", *Proceedings of the International Conference on Management of Data, (ACM SIGMOD'99)*, Vol. 28, pp. 49–60, ACM, 1999.

24. Bence, J. R., "Analysis of Short Time Series: Correcting for Autocorrelation", *Ecology*, Vol. 76, No. 2, pp. 628–639, 1995.

25. Arnau, J. and R. Bono, "Autocorrelation and Bias in Short Time Series: An Alternative Estimator", *Quality & Quantity*, Vol. 35, pp. 365–387, 2001.

26. Höppner, F., F. Klawonn, R. Kruse and T. Runkler, *Fuzzy Cluster Analysis*, John Wiley & Sons, Inc., Chichester, England, 1999.

27. Ben-Dor, A., R. Shamir and Z. Yakhini, "Clustering Gene Expression Patterns",

Journal of Computational Biology, Vol. 6, No. 3-4, pp. 281–297, 1999.

28. Ji, X., J. Li-Ling and Z. Sun, "Mining Gene Expression Data Using a Novel Approach Based on Hidden Markov Models", *Federation of European Biochemical Societies Letters*, Vol. 542, No. 1-3, pp. 125–131, 2003.

29. Gasch, A. P. and M. B. Eisen, "Exploring the Conditional Coregulation of Yeast Gene Expression Through Fuzzy K-Means Clustering", *Genome Biology*, Vol. 3, No. 11, pp. 1–22, 2002.

30. Heyer, L. J., S. Kruglyak and S. Yooseph, "Exploring Expression Data: Identification and Analysis of Coexpressed Genes", *Genome research*, Vol. 9, No. 11, pp. 1106–1115, 1999.

31. Dave, R. N., "Characterization and Detection of Noise in Clustering", *Pattern Recognition Letters*, Vol. 12, No. 11, pp. 657–664, 1991.

32. Fraley, C. and A. E. Raftery, "How Many Clusters? Which Clustering Method? Answers via Model-Based Cluster Analysis", *The Computer Journal*, Vol. 41, No. 8, pp. 578–588, 1998.

33. Yeung, K. Y., C. Fraley, A. Murua, A. E. Raftery and W. L. Ruzzo, "Model-Based Clustering and Data Transformations for Gene Expression Data", *Bioinformatics*, Vol. 17, No. 10, pp. 977–987, 2001.

34. Rao, S. T., E. Prasad and N. Venkateswarlu, "A Critical Performance Study of Memory Mapping on Multi-Core Processors: An Experiment With K-Means Algorithm With Large Data Mining Data Sets", *International Journal of Computer Applications*, Vol. 1, No. 9, pp. 90–98, 2010.

35. Ngai, W. K., B. Kao, C. K. Chui, R. Cheng, M. Chau and K. Y. Yip, "Efficient Clustering of Uncertain Data", *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pp. 436–445, IEEE, 2006.

36. Chatzis, S. P., "A Fuzzy C-Means-Type Algorithm for Clustering of Data With Mixed Numeric and Categorical Attributes Employing a Probabilistic Dissimilarity Functional", *Expert Systems With Applications*, Vol. 38, pp. 8684–8689, 2011.

37. Izakian, H. and A. Abraham, "Fuzzy C-Means and Fuzzy Swarm for Fuzzy Clustering Problem", *Expert Systems With Applications*, Vol. 38, pp. 1835–1838, 2011.

38. Li, D., H. Gu and L. Zhang, "A Fuzzy C-Means Clustering Algorithm Based on Nearest-Neighbor Intervals for Incomplete Data", *Expert Systems With Applications*, Vol. 37, pp. 6942–6947, 2010.

39. Troyanskaya, O. and M. Cantor, "Missing Value Estimation Methods for DNA Microarrays", *Bioinformatics*, Vol. 17, pp. 520–525, 2001.

40. Aach, J. and G. M. Church, "Aligning Gene Expression Time Series With Time Warping Algorithms", *Bioinformatics*, Vol. 17, pp. 495–508, 2001.

41. D'haeseleer, P., X. Wen, S. Fuhrman and R. Somogyi, "Linear Modeling of mRNA Expression Levels During CNS Development and Injury.", *Pacific Symposium on Biocomputing*, Vol. 4, pp. 41–52, 1999.

42. Bar-Joseph, Z., "Analyzing Time Series Gene Expression Data", *Bioinformatics*, Vol. 20, No. 16, pp. 2493–2503, 2004.

43. Sivriver, J., N. Habib and N. Friedman, "An Integrative Clustering and Modeling Algorithm for Dynamical Gene Expression Data", *Bioinformatics*, Vol. 27, pp. 1392–1400, 2011.

44. Liu, X., K. K. Lin, B. Andersen and M. Rattery, "Including Probe-Level Uncertainty in Model-Based Gene Expression Clustering", *BMC Bioinformatics*, Vol. 8, p. 98, 2007, `http://www.biomedcentral.com/1471-2105/8/98`.

45. Subhani, N., L. Rueda, A. Ngom and C. J. Burden, "Clustering Microarray Time-

Series Data Using Expectation Maximization and Multiple Profile Alignment", *Bioinformatics and Biomedicine Workshop, 2009. BIBMW 2009. IEEE International Conference on*, pp. 2–7, IEEE, 2009.

46. Ma, P. C. H., K. C. C. Chan, Y. Xin and D. K. Y. Chiu, "An Evolutionary Clustering Algorithm for Gene Expression Microarray Data Analysis", *IEEE Transactions on Evolutionary Computation*, pp. 296–314, 2006.

47. Holter, N. S., A. Maritan, M. Cieplak, N. V. Fedoroff and J. R. Banavar, "Dynamic Modeling of Gene Expression Data", *Proceedings of the National Academy of Sciences*, Vol. 98, No. 4, pp. 1693–1698, 2001.

48. Zhao, L. P., R. Prentice and L. Breeden, "Statistical Modeling of Large Microarray Data Sets to Identify Stimulus-Response Profiles", *Proceedings of the National Academy of Sciences*, Vol. 98, No. 10, pp. 5631–5636, 2001.

49. Eisen, M. B., P. T. Spellman, P. O. Brown and D. Botstein, "Cluster Analysis and Display of Genome-Wide Expression Patterns", *Proceedings of the National Academy of Sciences*, Vol. 95, No. 25, pp. 14863–14868, 1998.

50. Lein, E. S., M. J. Hawrylycz, N. Ao, M. Ayres, A. Bensinger, A. Bernard, A. F. Boe, M. S. Boguski, K. S. Brockway, E. J. Byrnes *et al.*, "Genome-Wide Atlas of Gene Expression in the Adult Mouse Brain", *Nature*, Vol. 445, No. 7124, pp. 168–176, 2007.

51. Bittner, M., P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor *et al.*, "Molecular Classification of Cutaneous Malignant Melanoma by Gene Expression Profiling", *Nature*, Vol. 406, No. 6795, pp. 536–540, 2000.

52. Sørlie, T., R. Tibshirani, J. Parker, T. Hastie, J. Marron, A. Nobel, S. Deng, H. Johnsen, R. Pesich, S. Geisler *et al.*, "Repeated Observation of Breast Tumor Subtypes in Independent Gene Expression Data Sets", *Proceedings of the National*

Academy of Sciences, Vol. 100, No. 14, pp. 8418–8423, 2003.

53. Tibshirani, R., T. Hastie, B. Narasimhan and G. Chu, "Diagnosis of Multiple Cancer Types by Shrunken Centroids of Gene Expression", *Proceedings of the National Academy of Sciences*, Vol. 99, No. 10, pp. 6567–6572, 2002.

54. Goldberg, M. K., M. Hayvanovych and M. Magdon-Ismail, "Measuring Similarity Between Sets of Overlapping Clusters", *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pp. 303–308, IEEE, 2010.

55. Möller-Levet, C., K. Cho, H. Yin and O. Wolkenhauer, *Clustering of Gene Expression Time-Series Data*, Tech. rep., Department of Computer Science, University of Rostock, Germany, 2003.

56. Brown, P., *Cell Cycle Regulated Yeast Genes*, 2007, `http://genome-www.stanford.edu/cellcycle/data/rawdata/individual.html`, accessed at June 2014.

57. Wu, W., M. Zou, D. R. Brickley, T. Pew and S. D. Conzen, "Glucocorticoid Receptor Activation Signals Through Forkhead Transcription Factor 3a in Breast Cancer Cells", *Molecular Endocrinology*, Vol. 20, No. 10, pp. 2304–2314, 2006.

58. Zou, M., W. Wu, D. R. Brickley, T. Pew and S. D. Conzen, *Time Course Microarray Data Following GR Activation in MCF10A-Myc Breast Cells*, 2006, `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE4917`, accessed at June 2014.