People's Democratic Republic Of Algeria

Ministry of Higher Education and Scientific Research

University Mohamed Boudiaf - M'sila

Faculty of Mathematics and Informatics

Department of Computer Science

Dissertation submitted in partial fulfillment of the requirements for the degree of MASTER

Domain: Mathematics and Computer Science

Field: Computer Science

Option: SIGL

By: Almoaatassam Bellah BENCHELLALI

Subject

Biclustering of Biological data

Publicly defended on: 13/07/2019 before the jury composed of:

Mohamed KAMEL	**University of M'sila**	**Chair**
Nasereddine AMROUNE	**University of M'sila**	**Supervisor**
Makhlouf BENAZI	**University of M'sila**	**Examiner**

Academic Year: 2018 /2019

ACKNOWLEDGEMENT

I would first like to thank my dissertation advisor Mr. Nasereddine AMROUNE. The door to Mr. AMROUNE office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought i needed it.

Finally, I must express my very profound gratitude to my parents, my brother, and my little sister for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

CHAPTER I : BIOLOGICAL DATA

CHAPTER II : BICLUSTERING

Chapter III: Proposed Approach

GENERAL INTRODUCTION

Microarrays enables the collection of vast amounts of gene expression data from biological systems. A single microarray chip can collect expression levels from thousands of genes, and this data is often collected from multiple tissues, in multiple patients, with different medical conditions, at different times, and in multiple trials. The knowledge that genes co-regulate with one another under a particular set of experimental conditions is demanded because this provides information for scientists to determine the genes that participate in the same biological processes.

Clustering of data sets techniques is utilized in a number of applications. In particular, biclustering is very important in the field of the gene expression data since genes may only jointly respond over a subset of conditions. Biclustering algorithms also have important applications in sample classification where, for instance, tissue samples can be classified as cancerous or normal. Many of the methods for biclustering, and clustering algorithms in general, utilize simplified models or heuristic strategies for identifying the "best" grouping of elements according to some metric, cluster definition, and thus result in suboptimal clusters.

Clustering can be accomplished based on genes, samples, and/or time variable, depending on the type of dataset, the performance of most algorithms today degraded as the number of biclusters in the dataset increased. This is especially a concern for large gene expression datasets, which may contain hundreds of biclusters. Because of the huge and growing data, we are proposed model to reduce time and get better result with big data, intuitive and fast.

The classical clustering methods fail to extract these relations because they are only able to either partition genes using the whole column data or cluster columns using the whole row data. Recently, biclustering is investigated to discover the underlined relations. The biclustering approaches can group a subset of genes in a subset of experimental conditions with respect to some similarity scores.

Though the popularity of biclustering has led to the publication of many algorithms, there are still many open lines of research. The landscape of possible bicluster models is relatively unexplored, in particular, it is not clear which models produce the most biologically relevant biclusters. Many new algorithms are possible, and existing algorithms may be modified to improve their results. New methods are published frequently, so the number of available algorithms from which to choose is expanding. As the number of available methods grows, the problem of selecting the appropriate algorithm for a data-mining task becomes increasingly difficult.

In this dissertation, we will propose a new model algorithm or modification in an original algorithm, explain the proposed model and develop a plan to install it, working on the Python environment, displaying results of different real and artificial datasets microarray, finally we have to discuss the results.

The dissertation is organized in three chapters:

- Chapter 1 Biological data: Introducing Bioinformatics field and molecular structures.
- Chapter 2 Biclustering: Overview of Clustering and Biclustering.
- Chapter 3 Proposed approach: explain and evaluate the performance of our proposed model.

CHAPTER 1
BIOLOGICAL DATA

CHAPTER 1 BIOLOGICAL DATA

1 Introduction

Biological data and DNA sequence data in particular, are accumulating at a phenomenal rate. By around 2005, it is likely that the DNA sequence of the complete human genome will have been determined. Although this achievement might seem an end in itself, in reality it is only the beginning. In order to exploit the wealth of DNA sequence and other biological data, a new science has arisen that fuses biology with mathematics and computer science called 'Bioinformatics'.

Bioinformatics is by nature a cross-disciplinary field that began in the 1960s with the efforts of Margaret O. Dayhoff, Walter M. Fitch, Russell F. Doolittle and others and has matured into a fully developed discipline. However, bioinformatics is wide-encompassing and is therefore difficult to define. For others, it is plainly computational science applied to a biological system. Bioinformatics is also a thriving field that is currently in the forefront of science and technology. Our society is investing heavily in the acquisition, transfer and exploitation of data and bioinformatics is at the center stage of activities that focus on the living world. It is currently a hot commodity [1].

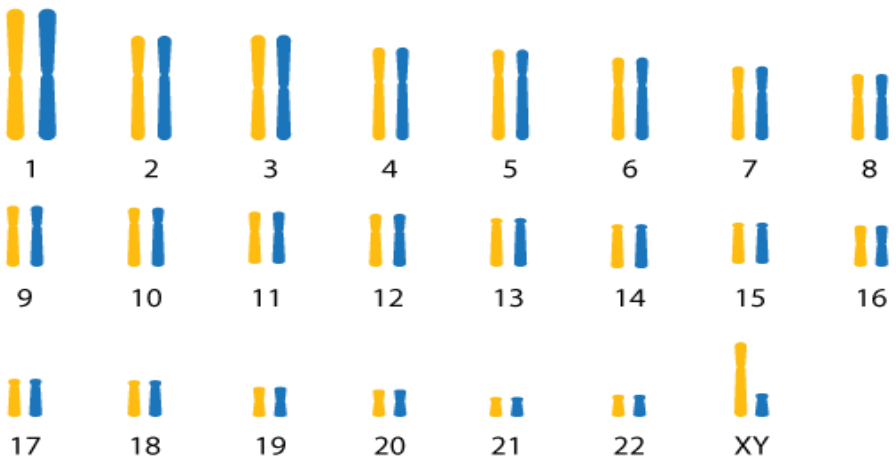
Twenty-first century biology will be a data-intensive enterprise. They will provide confirming or disconfirming evidence for the various theories and models of biological phenomena that researchers build. also, because 21st century biology will be a collective effort, it is critical that data be widely shareable and interoperable among diverse laboratories and computer systems. This chapter describes the nature of biological data and the requirements that scientists place on data so that they are useful [2].

The collect of biological data requires the use of computer science, mathematical, science, and biology principles. It is in the cross roads of experimental data and theoretical data. It is about understanding the molecular world. It is truly inter-disciplinary and is changing. Is moving from basic data to big data.

1.1 Important biology molecules:

1.1.1 The Genome

The hereditary information that an organism passes to its offspring is represented in each of its cells. The representation is in the form of DNA molecules.

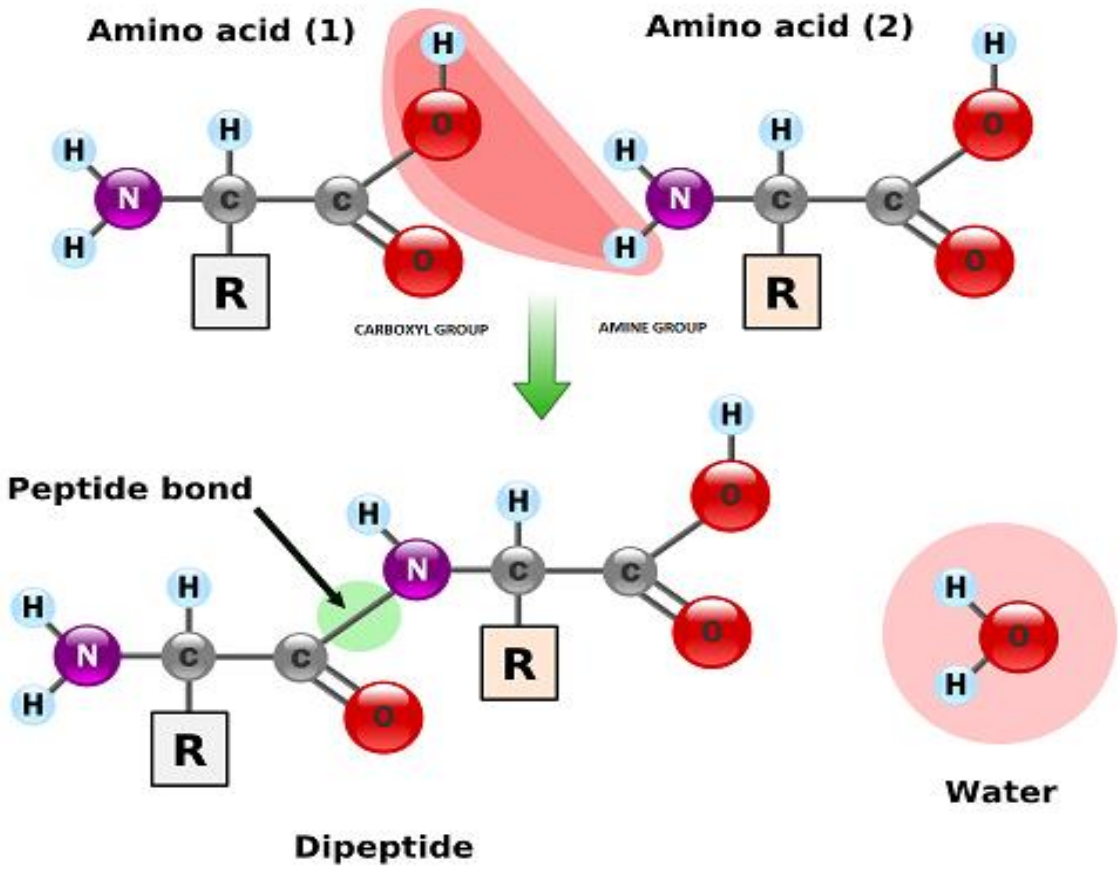


➢ **Figure 1. 1 A conceptual of Genome**

Each DNA molecule is a long chain of chemical structures called nucleotides of four different types, which can be viewed abstractly as characters from the alphabet A; C; T; G. The totality of this information is called the genome of the organism. In humans, the genome consists of nucleotides. A major task of molecular biology is to Extract the information contained in the genomes of different organisms; Elucidate the structure of the genome; Apply this knowledge to the diagnosis and ultimately, treatment, of genetic diseases (about 4000 such diseases in humans have been identified); By comparing the genomes of different species, explain the process and mechanisms of evolution. These tasks require the invention of new algorithms. [1]

1.1.2 Proteins

A protein is a very large biological molecule composed of a chain of smaller molecules called amino acids. Thousands of different proteins are present in a cell, the synthesis of each type of protein being directed by a different gene. There are 21 amino acids found in humans. Different combinations of these amino acids make up all of the proteins you can think of, Proteins make up much of the cellular structure (our hair, skin, and fingernails consist largely of protein. [3]

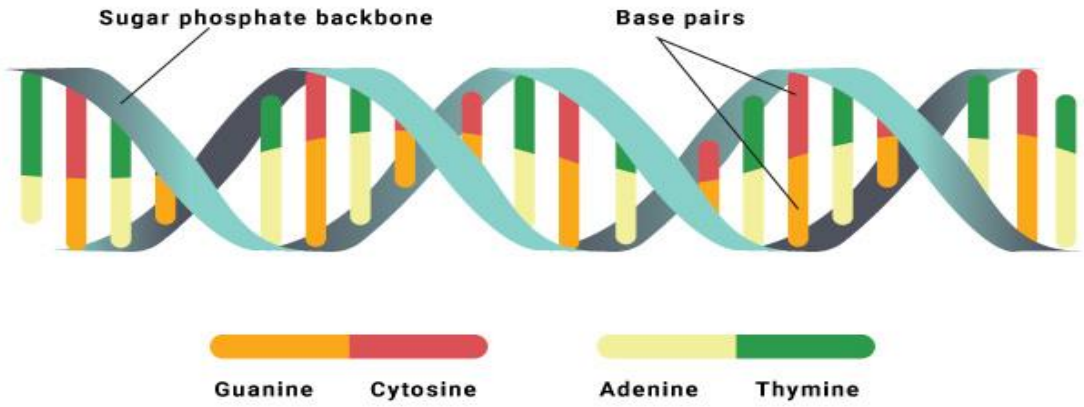


➢ **Figure 1. 2 Protein Structure**

1.1.3 DNA

DNA was discovered in 1869. Most of the DNA in cells is contained in the chromosomes. DNA is chemically very different from protein. DNA is structured as a double helix consisting of two long strands that wind around a common axis. Each strand is a very long chain of nucleotides of four types, A, C, T and G. There are four different types of nucleotides, distinguished by rings called bases, together with a common portion consisting of a sugar called deoxyribose and a phosphate group. On the sugar there are two sites, called the 3' and 5'

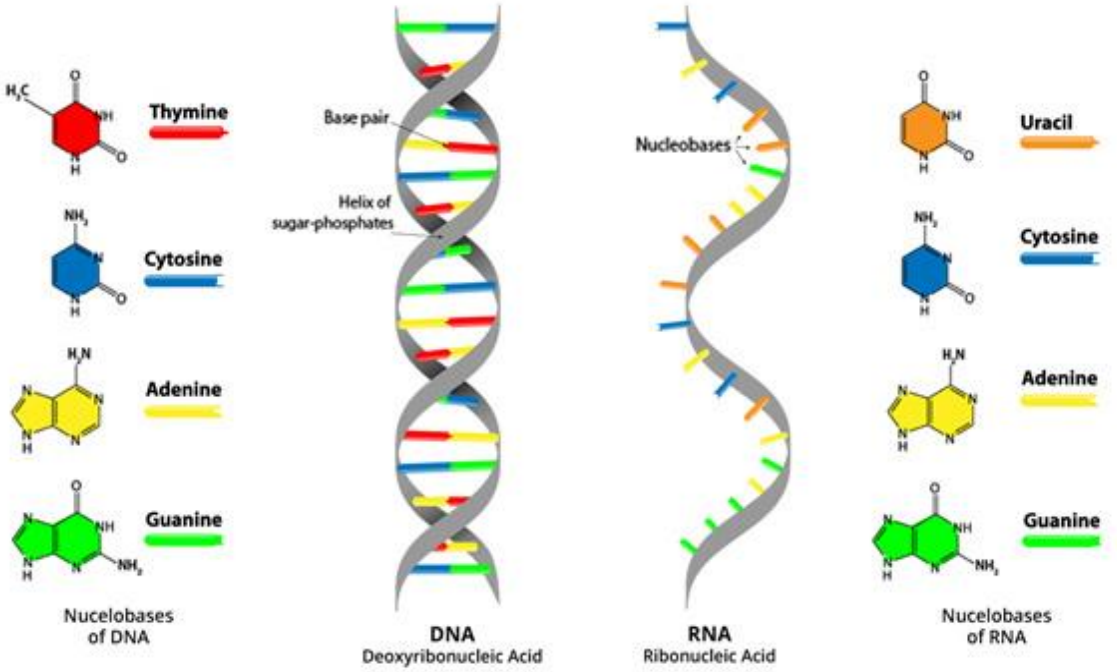
sites. Each phosphate is bonded to two successive sugars, at the 3' site of one and the 5' site of the other. These phosphate-sugar links form the backbones of the two chains. The bases of the two chains are weakly bonded together in complementary pairs each of the form CG or AT. Thus the chains have directionality (from 3' to 5'), and the sequence of bases on one chain determines the sequence on the other, by the rule of complementarity. [3]



➢ **Figure 1. 3 The structure of DNA, the double-stranded helix**

1.1.4 RNA

RNA, abbreviation of Ribonucleic Acid, complex compound of high molecular weight that functions in cellular protein synthesis and replaces DNA (deoxyribonucleic acid) as a carrier of genetic codes in some viruses. RNA consists of ribose nucleotides (nitrogenous bases appended to a ribose sugar) attached by phosphodiester bonds, forming strands of varying lengths. The nitrogenous bases in RNA are adenine, guanine, cytosine, and uracil, which replaces thymine in DNA. [3]



➢ **Figure 1. 4 The structure of DNA vs RNA**

RNA typically is a single-stranded biopolymer. However, the presence of self-complementary sequences in the RNA strand leads to intrachain base-pairing and folding of the ribonucleotide chain into complex structural forms consisting of bulges and helices. The three-dimensional structure of RNA is critical to its stability and function.

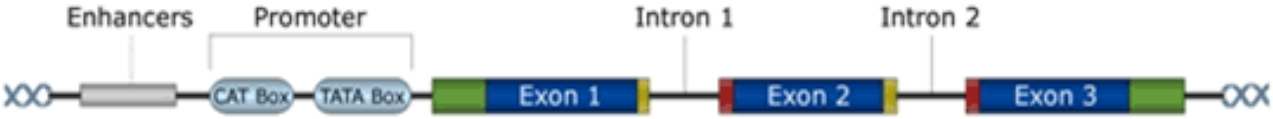
Of the many types of RNA, the three most well-known and most commonly studied are messenger RNA (mRNA), transfer RNA (tRNA), and ribosomal RNA (rRNA), which are present in all organisms. These and other types of RNAs primarily carry out biochemical reactions, similar to enzymes. Some, however, also have complex regulatory functions in cells. Owing to their involvement in many regulatory processes, to their abundance, and to their diverse functions, RNAs play important roles in both normal cellular processes and diseases. [3]

1.2 Gene **expression**

Gene expression is the process by which the genetic code – the nucleotide sequence – of a gene is used in the synthesis of a functional gene product the process of gene expression involves two main stages: transcription, translation. [3]

Some genes are responsible for the production of other forms of RNA that play a role in translation, including transfer RNA (tRNA) and ribosomal RNA (rRNA).

A structural gene involves a number of different components:



➢ **Figure 1. 5 Gene Structure**

Exons: Exons code for amino acids and collectively determine the amino acid sequence of the protein product. It is these portions of the gene that are represented in final mature mRNA molecule.

Introns: Introns are portions of the gene that do not code for amino acids, and are removed (spliced) from the mRNA molecule before translation.

1.2.1 Gene regulation

All cells contain exactly the same genetic material and yet some specialize as skin cells, muscle cells and so on. Cell specialization occurs because cells are able to express, or turn on, only a fraction of their genes while the others are turned off. The process of turning genes on and off is known as gene regulation. [4]

Gene regulation is an important part of normal development. Genes are turned on and off in a coordinated fashion during development in order to make cells look and function as specialized cells, such as brain or nerve cells. [4]

Signals from the environment or from other cells activate proteins called transcription factors which bind to regions of DNA close to the gene that attract proteins and literally switch the gene on or off by inhibiting or attracting the molecular machinery that allows for gene expression. [4]

But the control of genes is a dynamic chemical process reacting to many signals to ensure the healthy function of the cell.

1.2.2 Gene control regions

- **Start site:** A start site for transcription.
- **A promoter:** A region a few hundred nucleotides 'upstream' of the gene (toward the 5' end). It is not transcribed into mRNA, but plays a role in controlling the transcription of the gene. Transcription factors bind to specific nucleotide sequences in the promoter region and assist in the binding of RNA polymerases.
- **Enhancers:** Some transcription factors (called activators) bind to regions called 'enhancers' that increase the rate of transcription. These sites may be thousands of nucleotides from the coding sequences or within an intron. Some enhancers are conditional and only work in the presence of other factors as well as transcription factors.
- **Silencers:** Some transcription factors (called repressors) bind to regions called 'silencers' that depress the rate of transcription.

1.2.3 Transcription

Transcription is the process of RNA synthesis, controlled by the interaction of promoters and enhancers. Several different types of RNA are produced, including messenger RNA (mRNA), which specifies the sequence of amino acids in the protein product, plus transfer RNA (tRNA) and ribosomal RNA (rRNA), which play a role in the translation process.

Transcription involves four steps: [3]

1. **Initiation**: The DNA molecule unwinds and separates to form a small open complex. RNA polymerase binds to the promoter of the template strand.
2. **Elongation**: RNA polymerase moves along the template strand, synthesizing an mRNA molecule. In prokaryotes RNA polymerase is a holoenzyme consisting of a number of subunits, including a sigma factor (transcription factor) that recognizes the promoter. In eukaryotes there are three RNA polymerases: I, II and III. The process includes a proofreading mechanism.
3. **Termination**: In prokaryotes there are two ways in which transcription is terminated. In Rho-dependent termination, a protein factor called "Rho" is responsible for disrupting the complex involving the template strand, RNA polymerase and RNA molecule. In Rho-independent termination, a loop forms at the end of the RNA molecule, causing it to detach itself. Termination in eukaryotes is more complicated, involving the addition of additional adenine nucleotides at the 3' of the RNA transcript (a process referred to as polyadenylation).
4. **Processing**: After transcription the RNA molecule is processed in a number of ways: introns are removed and the exons are spliced together to form a mature mRNA molecule consisting of a single protein-coding sequence. RNA synthesis involves the normal base pairing rules, but the base thymine is replaced with the base uracil.

1.2.4 Translation

In translation the mature mRNA molecule is used as a template to assemble a series of amino acids to produce a polypeptide with a specific amino acid sequence. The complex in the cytoplasm at which this occurs is called a ribosome. Ribosomes are a mixture of ribosomal proteins and ribosomal RNA (rRNA), and consist of a large subunit and a small subunit.

Translation involves four steps: [3]

1. **Initiation:** The small subunit of the ribosome binds at the 5' end of the mRNA molecule and moves in a 3' direction until it meets a start codon (AUG). It then forms a complex with the large unit of the ribosome complex and an initiation tRNA molecule.

2. **Elongation:** Subsequent codons on the mRNA molecule determine which tRNA molecule linked to an amino acid binds to the mRNA. An enzyme peptidyl transferase links the amino acids together using peptide bonds. The process continues, producing a chain of amino acids as the ribosome moves along the mRNA molecule.
3. **Termination:** Translation in terminated when the ribosomal complex reached one or more stop codons (UAA, UAG, UGA). The ribosomal complex in eukaryotes is larger and more complicated than in prokaryotes. In addition, the processes of transcription and translation are divided in eukaryotes between the nucleus (transcription) and the cytoplasm (translation), which provides more opportunities for the regulation of gene expression.
4. **Post-translation processing of the protein**

1.3 History of Biological Data

Over a century ago, Biological data history started with an Austrian monk named *Gregor Mendel*. He is known as the *"Father of Genetics"*. He cross-fertilized different colors of the same species of flowers. He kept careful records of the colors of flowers that he cross-fertilized and the color(s) of flowers they produced. *Mendel* illustrated that the inheritance of traits could be more easily explained if it was controlled by factors passed down from generation to generation. Since *Mendel*, Biological data have come a long way. It has advanced remarkably in the last thirty years. [1]

In 1972, *Paul berg* made the first recombinant DNA molecule using ligase. In that same year, *Stanley Cohen*, *Annie Chang* and *Herbert Boyer* produced the first recombinant DNA organism. Then in 1973, two important things happened in the field of genomics, first one is *Joseph Sambrook* led a team that refined DNA electrophoresis using agarose gel, and second one was *Herbert Boyer* and *Stanely Cohen* invented DNA cloning, after that in 1977, a method for sequencing DNA was discovered and the first genetic engineering company, *Genetech* was founded. [1]

By 1981, 579 human genes had been mapped and mapping by insitu hybridization had become a standard method. *Marvin Carruthers* and *Leory Hood* made a huge leap in bioinformatics when they invented a method for automated DNA sequencing. The Human Genome organization (HUGO) foundation was in 1988, this is an international organization of scientists involved in Human Genome Project. [1]

The first complete genome map was published of the bacteria *Haemophilus* influenza was in 1989. The following year, the Human Genome Project was started, a total of 1879 human genes had been mapped in 1991. *Genethon*, a human genome research center in France Produced a physical map of the human genome. Three years later, *Genethon* published the final version of the Human Genetic Map. This concluded the end of the first phase of the Human Genome Project. [1]

Ten years ago, the only way to track genes was to scour large, well documented family trees of relatively inbred populations, such as the *Ashkenzai Jews* from Europe. These types of genealogical searches 11 million nucleotides a day for its corporate clients and company research. Bioinformatics was fueled by the need to create huge databases, such as *GenBank* and *EMBL* and DNA Database of Japan to store and compare the DNA sequence data erupting from the human genome and other genome sequencing projects. [1]

Today, bioinformatics embraces protein structure analysis, gene and protein functional information, data from patients, pre-clinical and clinical trials, and the metabolic pathways of numerous species.

2 Data Sources

Biology, like any science, changes when technology introduces new tools that extend the scope and type of inquiry. Some changes, such as the use of the microscope, are embraced quickly and easily, because they are consonant with existing values and practices. Others, such as the introduction of multivariate statistics as performed by computers in the 1960s, are resisted, because they go against traditions of intuition, visualization, and conceptions of biology that separate it clearly from mathematics. [2]

An immense challenge one of the most central facing 21st century biology is managing the variety and complexity of data types, the hierarchy of biology, and the inevitable need to acquire data by a wide variety of modalities. Biological data come in many types. For instance, biological data may consist of the following: [2]

- Microarrays
- Sequences (DNA, RNA, Protein)
- Structures of biological molecules
- Gene expression profiles
- Biochemical information
- Chromosomal mapping
- genomes
- gene expression
- Protein-protein interactions, complexes
- Phylogenetic data
- Single Nucleotide Polymorphisms (SNPs)

2.1 Biological data bases

Biological databases are libraries of life sciences information, collected from scientific experiments, published literature, high-throughput experiment technology, and computational analysis. They contain information from research areas including genomics, proteomics, metabolomics, microarray gene expression, and phylogenetic. Information contained in biological databases includes gene function, structure, localization (both cellular and chromosomal), clinical effects of mutations as well as similarities of biological sequences and structures. [5]

Biological databases can be broadly classified into sequence and structure databases. Nucleic acid and protein sequences are stored in sequence databases and structure database only store proteins. These databases are important tools in assisting scientists to analyze and explain a host of biological phenomena from the structure of biomolecules and their interaction, to the whole metabolism of organisms and to understanding the evolution of species. This knowledge helps facilitate the fight against diseases, assists in the development of medications, predicting certain genetic diseases and in discovering basic relationships among species in the history of life. [5]

Biological knowledge is distributed among many different general and specialized databases. This sometimes makes it difficult to ensure the consistency of information. Integrative bioinformatics is one field attempting to tackle this problem by providing unified access. One solution is how biological databases cross-reference to other databases with accession numbers to link their related knowledge together. [5]

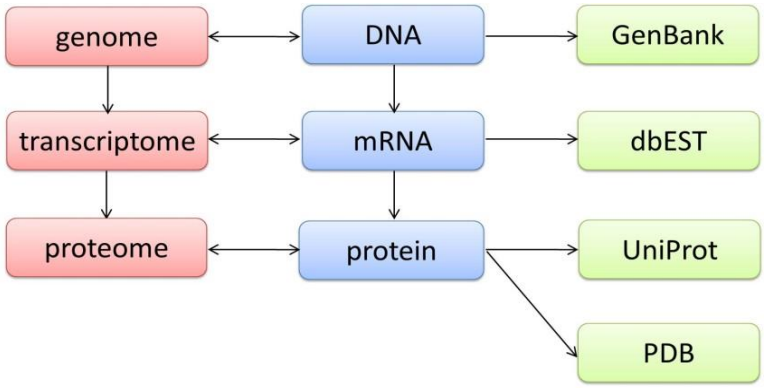
Relational database concepts of computer science and Information retrieval concepts of digital libraries are important for understanding biological databases. Biological database design,

development, and long-term management is a core area of the discipline of bioinformatics. Data contents include gene sequences, textual descriptions, attributes and ontology classifications, citations, and tabular data. These are often described as semi-structured data, and can be represented as tables, key delimited records, and XML structures.

Data bases Access

Most biological databases are available through web sites that organize data such that users can browse through the data online. In addition the underlying data is usually available for download in a variety of formats. Biological data comes in many formats. These formats include text, sequence data, protein structure and links. Each of these can be found from certain sources, for example: [6]

- Text formats are provided by PubMed and OMIM.
- Sequence data is provided by GenBank, in terms of DNA, and UniProt, in terms of protein.
- Protein structures are provided by PDB, SCOP, and CATH.



➢ **Figure 1. 6 Databases websites**

2.2 Experimental data

Biologists and other scientists use the scientific method to ask questions about the natural world. The scientific method begins with an observation, which leads the scientist to find new data.

Quantitative Observation

In the scientific method, after a scientist comes up with a theory based on an observation of something in nature. Once the experiment is underway, it must be observed. The scientist records the observations of the experiment and collects data. One form of data collection during the method is quantitative. This form of observation during an experiment employs mathematical models and relies on the scientist to collect information based on numbers, such as how many apples fell from a tree or balcony. Quantitative observation is common in physics, biology and the natural sciences. [7]

Qualitative Observation

When a scientist performs an experiment that requires observations concerning the quality of what has happened in an experiment, it is considered a qualitative observation or data. Examples include the shapes of the apples that fell from a balcony or tree or what happened to them when they fell. Qualitative observations can be easily dismissed in experiments that require hard mathematical data, but they are made nonetheless. Qualitative observations can be very important in experiments that require interpretation. [7]

3 Microarrays

DNA Microarray is a molecular technique used in scientific research for several purposes, such as the study of gene expression, the study of the effect of a drug on patients, etc. Many applications results of the test presented in the form of matrix illuminated by different colors by hybridization.

The principle of this technique depends on the hybridization of the material to be studied with thousands of genes on a small glass or plastic chip called a DNA chip. This chip contains many DNA fragments known as DNA probes. These probes are only a known and defined sequence of nucleotides and represent part of a particular gene. [8]

3.1 History of Microarrays

The original DNA array was created with the colony hybridization method of *Grunstein and Hogness*. In this method, the DNA of interest is cloned into Escherichia coli plasmids, and E. coli colonies with different hybrid plasmids can be screened to determine a specified DNA sequence or gene. DNA prints of the colonies are then hybridized to radioactive RNA, and are analyzed by autoradiography. This method can be used to isolate any gene. [8]

Using this approach, *Gergen al*. reported a method for making paper filter replicas of such an ordered collection and developed a strategy for creating a high-density (10,000 colonies/petri) unordered collection, these different mixtures of probes could be used for nucleic acid hybridization screens of recombinant DNA colonies. [8]

In 1980, *Crampton al*. compared RNA populations derived from normal human lymphocytes and fibroblasts by hybridizing each RNA to cDNA derived from the other RNA population. The isolation of cloned cDNA sequences revealed the differentially expression between two samples.

Schena et al. published a high-capacity system that was developed to analyze the gene expression in parallel. Microarray technologies which were prepared by high-speed robotic printing of complementary DNAs on glass were useful for quantitative expression analysis of the corresponding genes. Differential gene expression measurements were obtained using simultaneous, two-color fluorescence hybridization. [8]

In 1996, *DeRisi al.* published a method describing very high density cDNA microarrays on glass substrates using fluorescent probes, and these arrays were used to search for differences in gene expression associated with tumor suppression. [8]

Since these initial studies, DNA microarray technologies have developed rapidly in a variety of fields. In 2004 The entire human genome is successfully printed on one microarray.

3.2 Definition

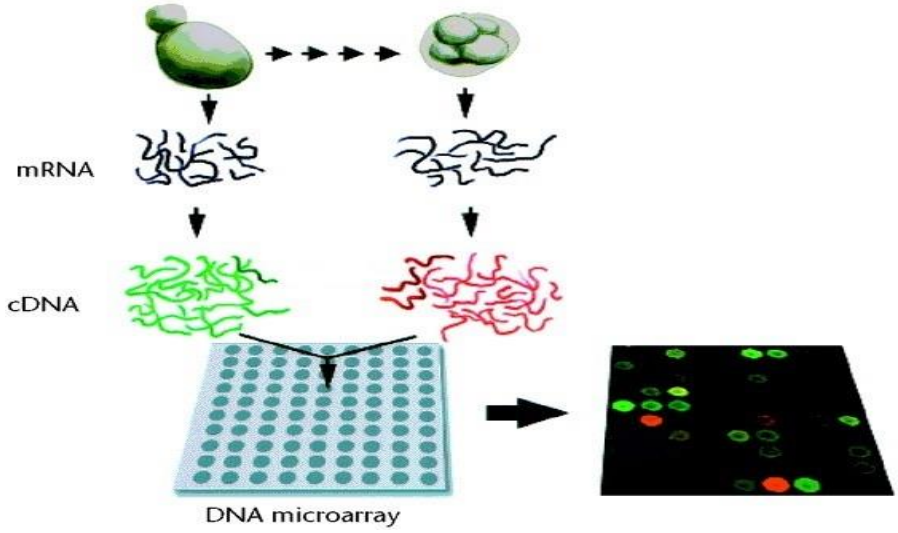
Microarrays are a two-dimensional arrangement of biological samples, a microarray may contain thousands of ‘spots’. Each spot contains many copies of the same DNA sequence that uniquely represents a gene from an organism. Spots are arranged in an orderly fashion into Pengroups. Schematic of the experimental protocol to study differential expression of genes. The organism is grown in two different conditions (a reference condition and a test condition). RNA is extracted from the two cells, and is labelled with different dyes (red and green) during the synthesis of cDNA by reverse transcriptase. Following this step, cDNA is hybridized onto the microarray slide, where each cDNA molecule representing a gene will bind to the spot containing its complementary DNA sequence. The microarray slide is then excited with a laser at suitable wavelengths to detect the red and green dyes. The final image is stored as a file for further analysis.

Gene expression matrices have been extensively analyzed in two dimensions: the gene dimension and the condition dimension. This correspond to the:

- Analysis of expression patterns of genes by comparing rows in the matrix.
- Analysis of expression patterns of samples by comparing columns in the matrix.

Common objectives pursued when analyzing gene expression data include:

1) Grouping of genes according to their expression under multiple conditions.
2) Classification of a new gene, given its expression and the expression of other genes, with known classification.
3) Grouping of conditions based on the expression of a number of genes.
4) Classification of a new sample, given the expression of the genes under that experimental condition.



➢ **Figure 1. 7 Microarray gene expression operation**

3.3 Types of Microarrays

Based on the types of probes used, microarrays are of twelve different types: [4]

1. **DNA microarrays:** DNA microarray is also known as gene chip, DNA chip, or biochip. It either measures DNA or uses DNA as a part of its detection system. There are four different types of DNA microarrays: cDNA microarrays, oligo DNA microarrays, BAC microarrays and SNP microarrays.
2. **MMChips:** MMchip allows the integrative analysis of cross-platform and between-laboratory data. It studies interactions between DNA and protein. ChIP-chip (Chromatin immunoprecipitation (ChIP) followed by array hybridization) and ChIP-seq (ChIP followed by massively parallel sequencing) are the two techniques used.
3. **Protein microarrays:** it acts as a platform for characterization of hundreds of thousands of proteins in a highly parallel way. Protein microarray is of three types, and these are analytical protein microarrays, functional protein microarrays and reverse-phase protein microarrays.
4. **Peptide microarrays:** these types of arrays are used for the detailed analyses or optimization of protein–protein interactions. It helps in antibody recognition by screening proteomes.
5. **Tissue microarrays:** tissue microarray paraffin blocks that are formed by separating cylindrical tissue cores from various donors and embedding it into a single microarray. This is mainly used in pathology.

6. **Cellular microarrays:** they are also called transfection microarrays or living-cell-microarrays, and are used for screening large-scale chemical and genomic libraries and systematically investigating the local cellular microenvironment.
7. **Chemical compound microarrays:** this is used for drug screening and drug discovery. This microarray has the capacity to identify and evaluate small molecules and so it is more useful than the other technologies used in the pharmaceutical industry.
8. **Antibody microarrays:** they are also referred to as antibody array or antibody chip. These are protein-specific microarrays that contain a collection of capture antibodies placed inside a microscope slide. They are used for detecting antigens.
9. **Carbohydrate arrays:** they are also called glycoarrays. Carbohydrate arrays are used in screening proteomes that are carbohydrate binding. They can also be utilized in calculating protein binding affinities and automization of solid-support synthesis for glycans.
10. **Phenotype microarrays:** phenotype microarrays or PMs are mainly used in drug development. They quantitatively measure thousands of cellular phenotypes all at once. It is also used in functional genomics and toxicological testing.
11. **Reverse phase protein microarrays:** they are microarrays of lysates or serum. Mostly used in clinical trials, especially in the field of cancer, they also have pharmaceutical uses. In some cases, they can also be used in the study of biomarkers.
12. **Interferometric reflectance imaging sensor or IRIS:** IRIS is a biosensor that is used to analyze protein–protein, protein–DNA, and DNA–DNA interactions. It does not make use of fluorescent labels. It is made of Si/SiO2 substrates prepared by robotic spotting.

3.4 DNA Microarrays Techniques

The first step in using a microarray is to collect healthy and cancerous tissue samples from the patient, this way, doctors can look at what genes are turned on and off in the healthy cells compared to the cancerous cells. Once the tissues samples are obtained, the messenger RNA (mRNA) is isolated from the samples. The mRNA is color-coded with fluorescent tags and used to make a DNA copy (the mRNA from the healthy cells is dyed green; the mRNA from the abnormal cells is dyed red.). The DNA copy that is made, called complementary DNA (cDNA), is then applied to the microarray. The cDNA binds to complementary base pairs in each of the spots on the array, a process known as hybridization. Based on how the DNA binds together, each spot will appear red, green, or yellow (a combination of red and green) when scanned with a laser. [9]

- A red spot indicates that that gene was strongly expressed in cancer cells. A green spot indicates that that gene was strongly repressed in cancer cells.
- If a spot turns yellow, it means that that gene was neither strongly expressed nor strongly repressed in cancer cells.
- A black spot indicates that none of the patient's cDNA has bonded to the DNA in the gene located in that spot. This indicates that the gene is inactive.

3.4.1 DNA microarray uses

When they were first introduced, DNA microarrays were used only as a research tool. Scientists continue today to conduct large-scale population studies - for example, to determine how often individuals with a particular mutation actually develop breast cancer, or to identify the changes in gene sequences that are most often associated with particular diseases. This has become possible

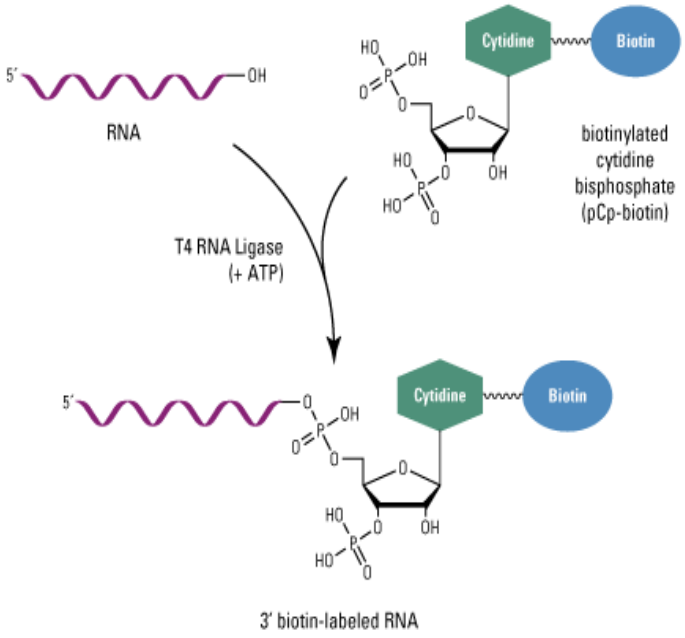
because, just as is the case for computer chips, very large numbers of 'features' can be put on microarray chips, representing a very large portion of the human genome.

Microarrays can also be used to study the extent to which certain genes are turned on or off in cells and tissues. In this case, instead of isolating DNA from the samples, RNA (which is a transcript of the DNA) is isolated and measured. [9]

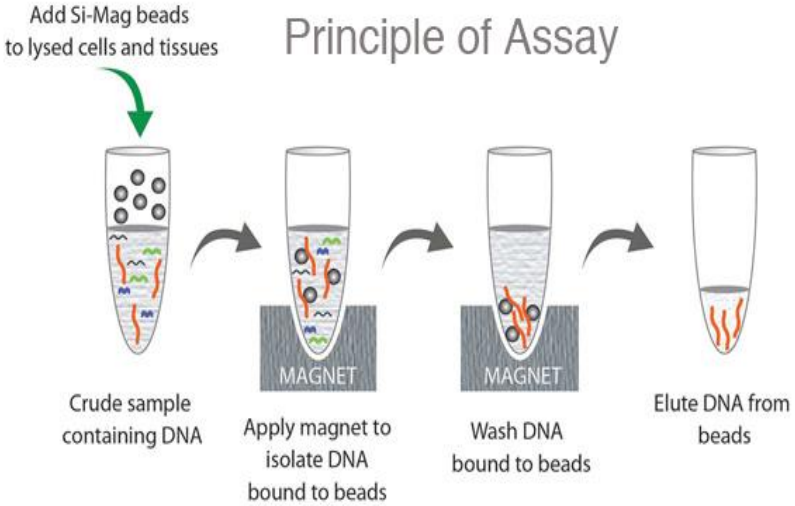
3.5 DNA Microarrays measures gene expression

A basic protocol for a DNA microarray is as follows: [3]

1) **Isolate and purify mRNA from samples of interest**: Since we are interested in comparing gene expression, one sample usually serves as control, and another sample would be the experiment (healthy vs. disease, etc.)

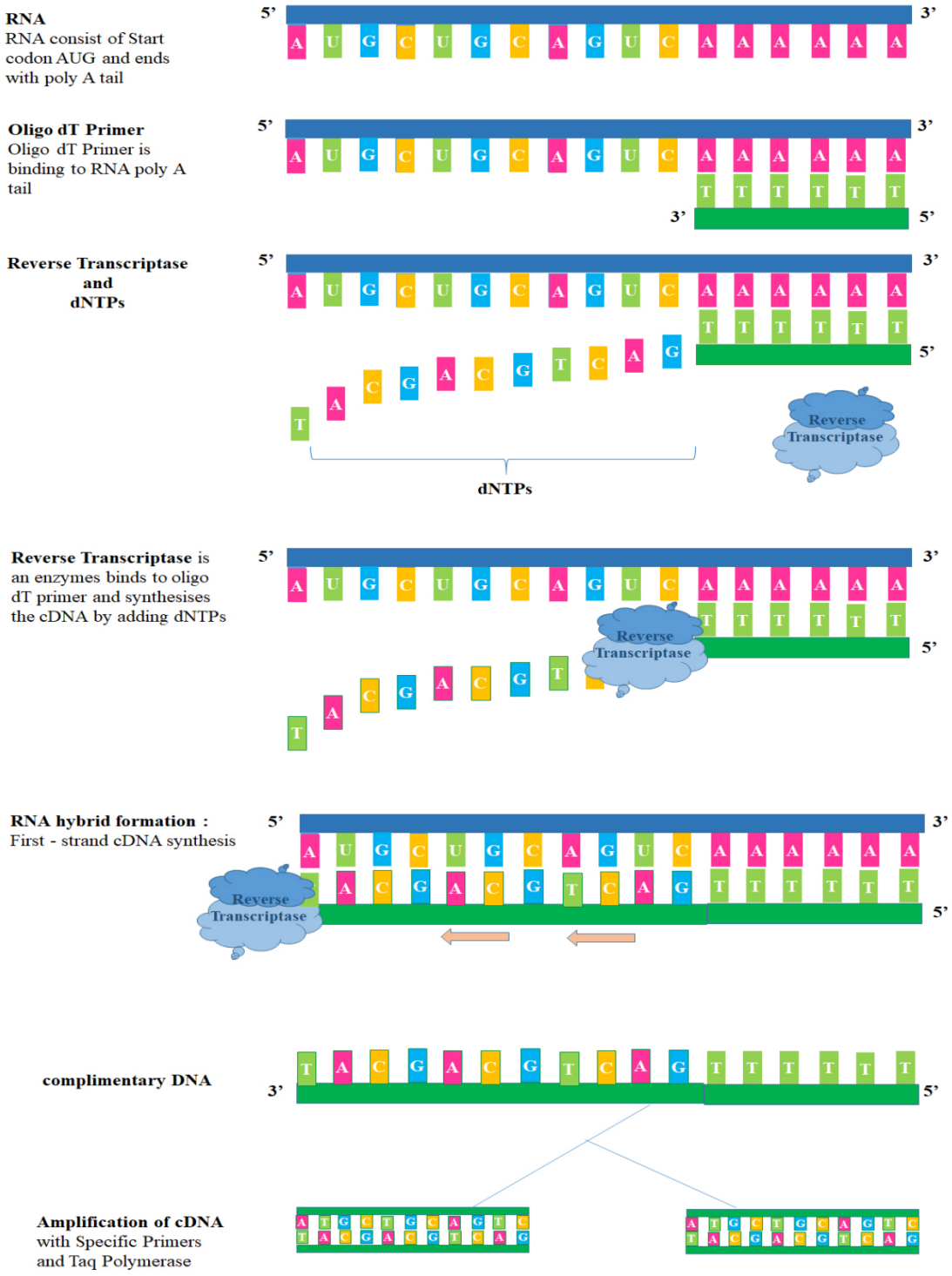


➢ **Figure 1. 8 Chemical process for labeling the 3′ end of RNA using T4 RNA**



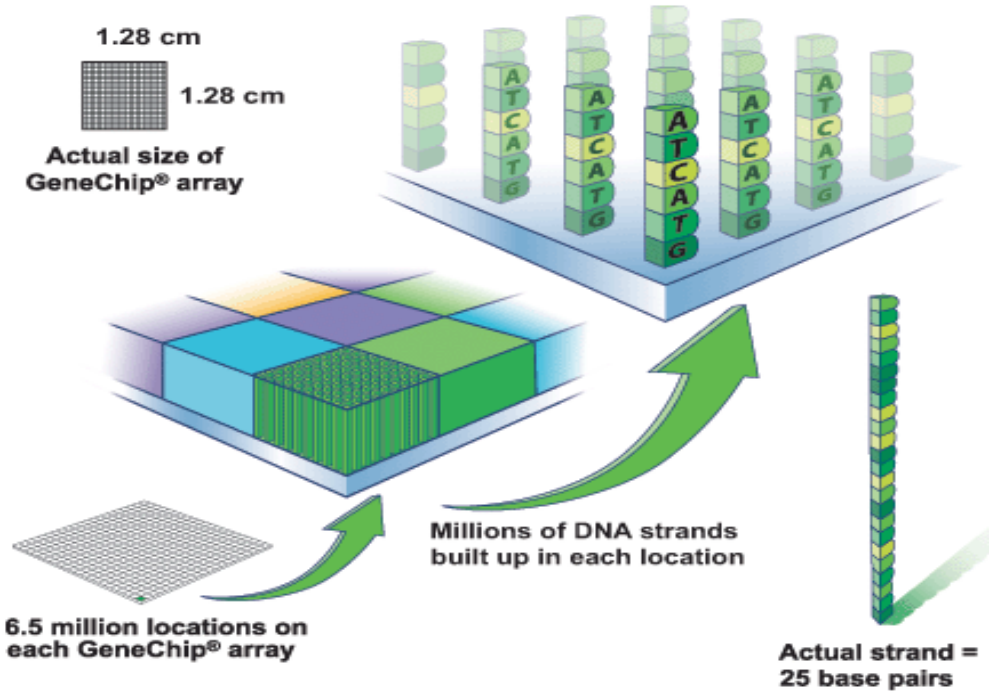
➢ **Figure 1. 9 Magnetic Beads for DNA Extraction and Purification**

2) **Reverse transcribe and label the mRNA**: In order to detect the transcripts by hybridization, they need to be labeled, and because starting material maybe limited, an amplification step is also used. Labeling usually involves performing a reverse transcription (RT) reaction to produce a complementary DNA strand (cDNA) and incorporating a florescent dye that has been linked to a DNA nucleotide, producing a fluorescent cDNA strand. Disease and healthy samples can be labeled with different dyes and co-hybridized onto the same microarray in the following step. Some protocols do not label the cDNA but use a second step of amplification, where the cDNA from RT step serves as a template to produce a labeled cRNA strand. [3]



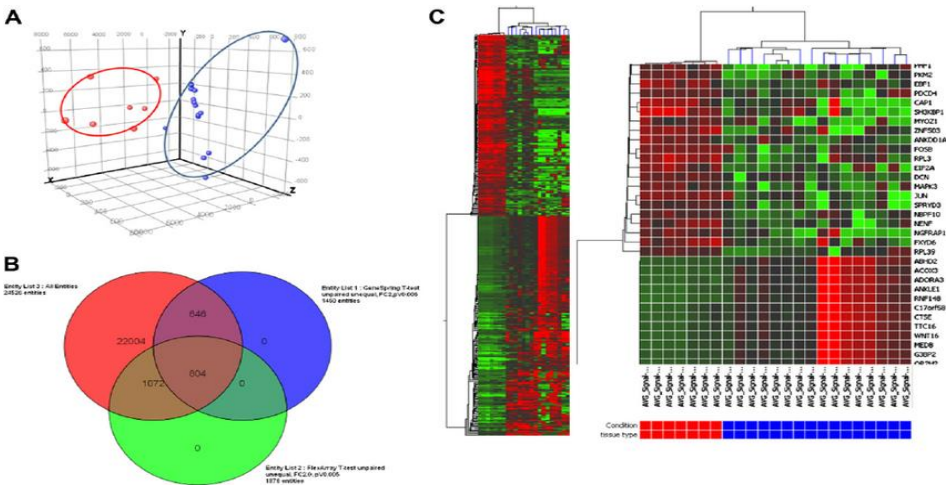
➢ **Figure 1. 10 Reverse transcription polymerase chain**

3) **Hybridize the labeled target to the microarray**: This step involves placing labeled cDNAs onto a DNA microarray where it will hybridize to their synthetic complementary DNA probes attached on the microarray. A series of washes are used to remove non-bound sequences. [3]



➢ **Figure 1. 11 Microarrays structure**

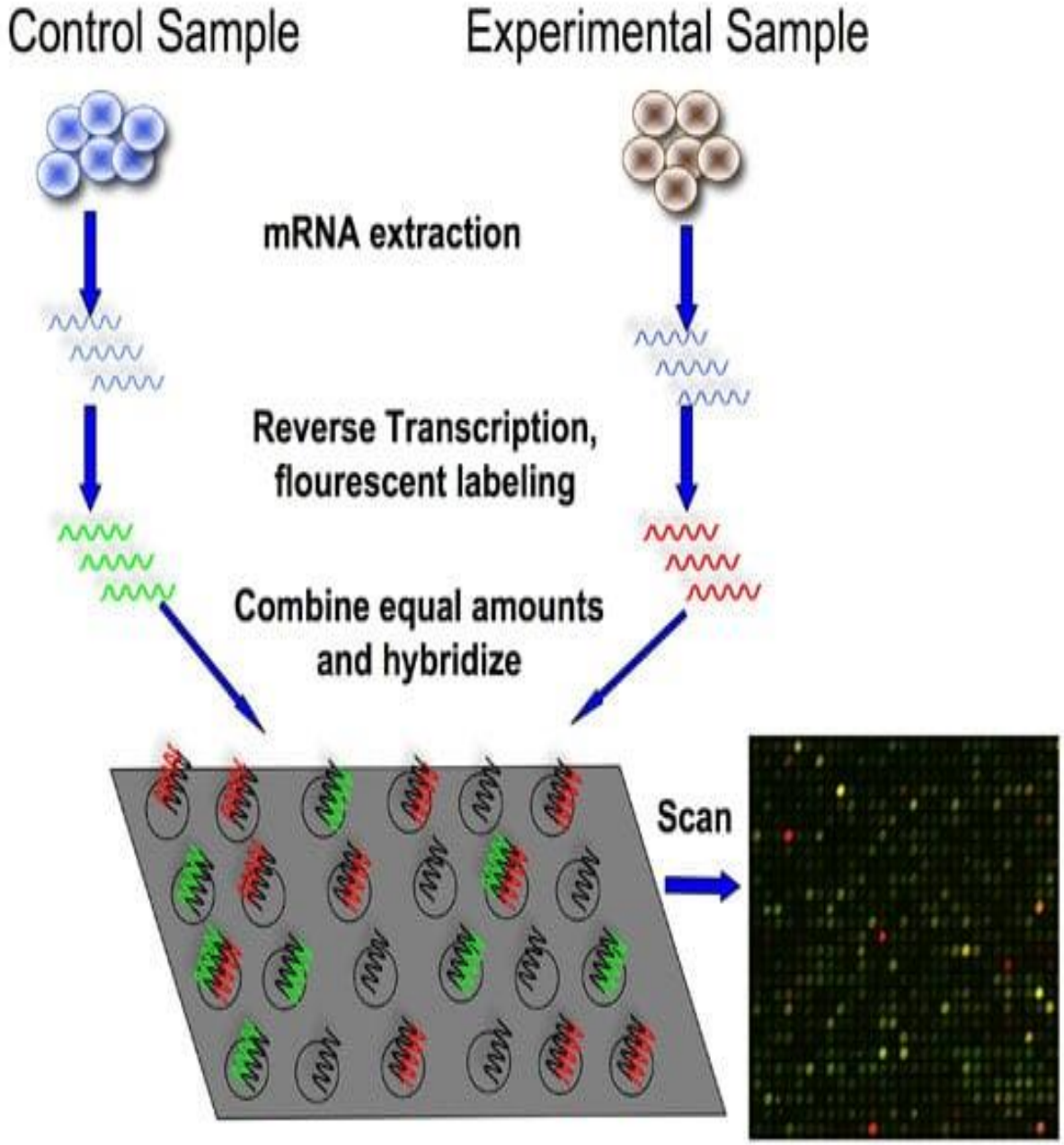
4) **Scan the microarray and quantitate the signal**: The fluorescent tags on bound cDNA are excited by a laser and the fluorescently labeled target sequences that bind to a probe generate a signal. The total strength of the signal depends upon the amount of target sample binding to the probes present on that spot. Thus, the amount of target sequence bound to each probe correlates to the expression level of various genes expressed in the sample. The signals are detected, quantified, and used to create a digital image of the array. [3]



➢ **Figure 1. 12 Microarray data analysis**

5) **Microarray data analysis**: Microarray data analysis is the final step in reading and processing data produced by a microarray chip, a large amount of data that requires processing via computer software. [3]

If we are trying to calculate relative expression between two samples, each labeled with a different dye (See figure 1.13, red for experiment, green for control), the resulting image is analyzed by calculating the ratio of the two dyes. If a gene is over-expressed in the experimental sample, then more of that sample cDNA than control cDNA will hybridize to the spot representing that expressed gene. In turn, the spot will fluoresce red with greater intensity than it will fluoresce green. The red-to-green fluorescence ratio thus indicates which gene is up or down regulated in the appropriate sample. [3]



➢ **Figure 1. 13 A typical DNA microarray co-hybridization (2 dye) experiment**

4 Conclusion

Today biology scientists are facing new challenges of analyzing massive amounts of data, we will need more powerful software agents to cluster, filter, organize and digest the information so that we can get on with the creative process of interpreting, describing and acting upon it. the most important technology to extract biological information is microarray.

Microarray data analysis is a revolution in scientific understanding, it helping biology scientists to gather more and more data about how cellular processes work. microarrays are a tool that allows a global vision, and made it possible to achieve a quantum leap in biological analysis. It is now possible to analyze the expression of thousands of genes in parallel and at high speed. similar developments are beginning parallel for massively analysis of proteins and other cellular components.

This tool opens up new research horizons to analyze the data obtained.

CHAPTER 2
BICLUSTERING

CHAPTER 2 BICLUSTERING

1 Introduction

The world today is drowning in a huge amount of data, the digital revolution has made digitized information easy to capture, process, store, distribute and transmit. The amount of data seems to go on and on increasing and the progress in digital data acquisition and storage technology has resulted in the growth of huge databases.

The knowledge Discovery from huge number of databases and massive volume of data is a challenge. Within these masses of data lies hidden information of strategic importance.

When there are so many trees, how do we draw meaningful conclusions about the forest? The newest answer is data mining, which is being used both to increase revenues and to reduce costs.

A large number of clustering approaches have been proposed for the analysis of gene expression data obtained from microarray experiments. However, the results of the application of standard clustering methods to genes are limited. These limited results are imposed by the existence of a number of experimental conditions where the activity of genes is uncorrelated. A similar limitation exists when clustering of conditions is performed.

For this reason, a number of algorithms that perform simultaneous clustering on the row and column dimensions of the gene expression matrix has been proposed to date. This simultaneous clustering, usually designated by biclustering, seeks to find sub-matrices, that is subgroups of genes and subgroups of columns, where the genes exhibit highly correlated activities for every condition. This type of algorithms has also been proposed and used in other fields, such as information retrieval and data mining. [10]

2 CLUSTERING

2.1 Definition

Data Clustering is unsupervised method which is used to group related data points in same cluster based on the similarity among data points. Clustering methods can be applied to either the rows or the columns of the data matrix, separately. Biclustering methods, on the other hand, perform clustering in the two dimensions simultaneously. This means that clustering methods derive a global model while biclustering algorithms produce a local model. When clustering algorithms are used, each gene in a given gene cluster is defined using all the conditions. Similarly, each condition in a condition cluster is characterized by the activity of all the genes. However, each gene in a bicluster is selected using only a subset of the conditions and each condition in a bicluster is selected using only a subset of the genes. The goal of biclustering techniques is thus to identify subgroups of genes and subgroups of conditions, by performing simultaneous clustering of both rows and columns of the gene expression matrix, instead of clustering these two dimensions separately. [10]

In biology, it can be used to define taxonomies, categorize genes with similar functionality and gain insights into structures inherent in populations.

2.2 Different Types of Clustering Algorithms

Well separated clusters are the clusters in which set of objects are significantly closer to each other than the objects which are not in the cluster. [11]

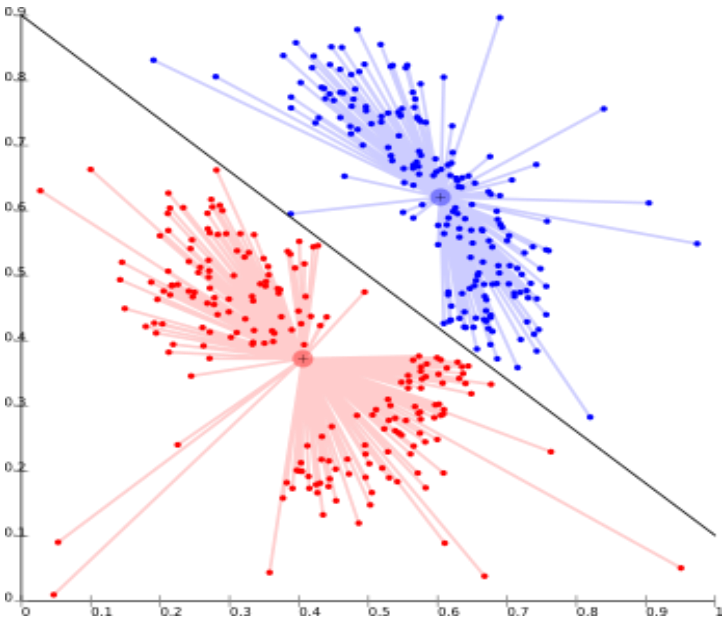
- Partitional algorithms
- Grid-based algorithms
- Hierarchical algorithms
- Density models

Partitional algorithmes

Partitioning clustering algorithm split the data points into ***k*** division, where each division represent a cluster and ***k*<=*n***, where n is the number of data points. Partitioning methods are based on the idea that a cluster can be represented by a centre point. The cluster must exhibit two properties: [11]

- Each collection should have at least one object.
- Every object should belong to accurately one collection.

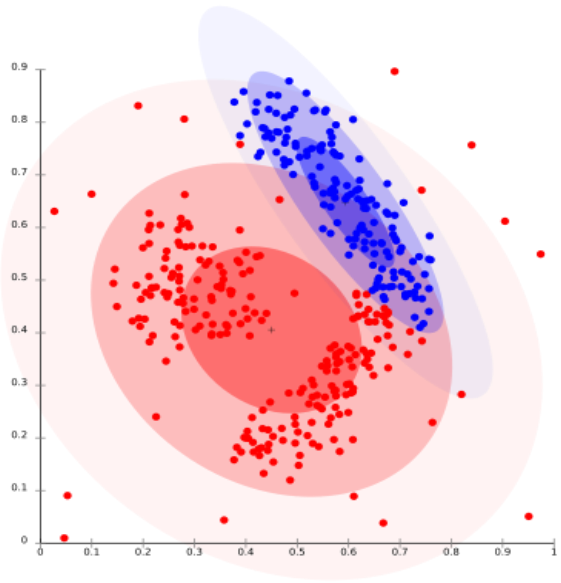
The main drawback of this algorithm is whenever a point is close to the center of another cluster; it gives poor outcome due to overlapping of data points.



➢ **Figure 2. 1 Partitional algorithms data visualization**

Grid-based algorithms

Grid-based clustering algorithms first cover the problem space domain with a uniform grid mesh. Statistical attributes are collected for all the data objects located in each individual mesh cell and clustering is then, performed on the grid, instead of data objects themselves. These algorithms typically have a fast processing time, since they go through the data set once to compute the statistical values for the grids and the performance of clustering depends only on the size of the grids which is usually much less than the data objects. [11]



➢ **Figure 2. 2 Grid-based algorithms data visualization**

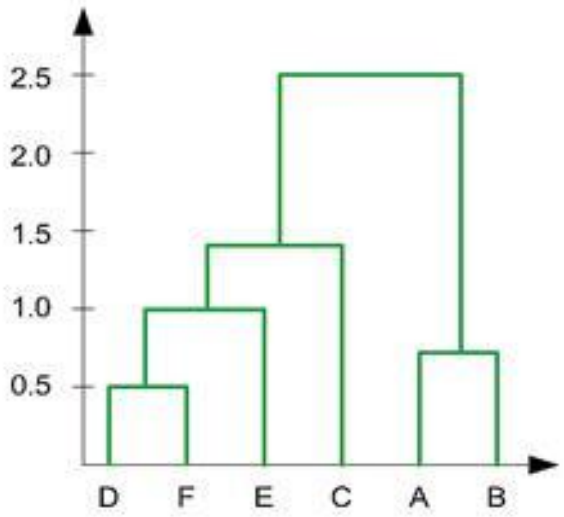
Hierarchical algorithms

Hierarchical clustering algorithms according to the method that produce clusters can further be divided into: [11]

• ***Agglomerative algorithms:*** They produce a sequence of clustering schemes of decreasing number of clusters at east step. The clustering scheme produced at each step results from the previous one by merging the two closest clusters into one.

• ***Divisive algorithms:*** These algorithms produce a sequence of clustering schemes of increasing number of clusters at each step. Contrary to the agglomerative algorithms, the clustering produced at each step results from the previous one by splitting a cluster into two.

It is not a single partitioning of the data set; instead it provides an extensive hierarchy of clusters that merge with each other at certain distances. Here the choice of distance function is subjective. These models are very easy to interpret but it lacks scalability.

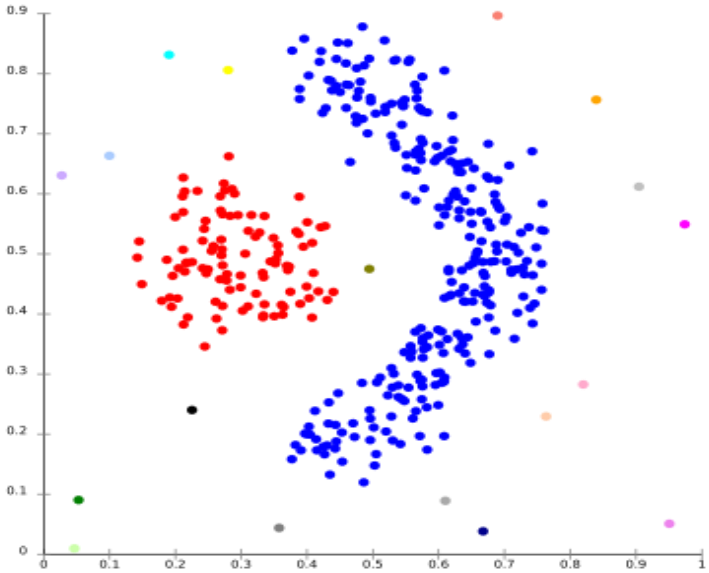


➢ **Figure 2. 3 Hierarchical algorithms data visualization**

Density models

In this clustering model there will be a searching of data space for areas of varied density of data points in the data space. It isolates various density regions based on different densities present in the data space. [11]

For Ex- DBSCAN and OPTICS.



➢ **Figure 2. 4 Density models data visualization**

2.3 Cluster validity measures

Normally, the user would guess the range of the number of clusters and then use the cluster validity measures to assess which particular number of clusters best reveals the true structure in the data. Cluster validity is a suite methodologies and algorithms that offer some mechanisms to validate clustering results. There are many measures called cluster validity indices, whose values relate to the number of clusters generated and thus are used to judge the clusters detected in the data and to assess the quality of the structure revealed in this manner.

2.3.1 Internal Criteria

As the name suggests, internal validation measures rely on information in the data only, that is the characteristics of the clusters themselves, such as compactness and separation. In the perfect world we want our clusters to be as compact and separated as possible. [12]

Connectivity

This measure reflects the extent to which items that are placed in the same cluster are also considered their nearest neighbors in the data space - or, in other words, the degree of connectedness of the clusters. it should be minimized. [12]

Silhouette Width

This index defines compactness based on the pairwise distances between all elements in the cluster, and separation based on pairwise distances between all points in the cluster and all points in the closest other cluster, we used silhouette function to assess the optimal number of clusters. [11]

Root Squared (RS) Index

Root Squared (RS) Index is aimed at measuring the dissimilarity of clusters. It is calculated by sum of squares between clusters to the total sum of squares of the whole data set.

The value of RS range from 0 to 1. If RS value is 0, it indicates there is no difference among clusters whereas 1 indicates clusters are considerably distinct. [12]

2.3.2 External **Criteria**

F-Measure

F-Measure is the harmonic mean of precision and recall values for each cluster. F-measure tries to balance the precision and recall values across all the clusters. The F-Measure values are within the interval [0, 1] and larger values indicate higher clustering quality. The maximum value of F-measure is thus one. [13]

NMI measure

NMI measure is called Normalized Mutual Information (NMI). Here the mutual information tries to quantify the amount of shared information between the clustering and the partition. The NMI value lies in the range [0, 1]. Values close to 1 indicate a good clustering. [13]

Entropy

Entropy measures the purity of the clusters class labels. The entropy value becomes zero if the object in a cluster have same class label. The entropy value increases when the class labels of objects in a cluster become more varied. For a perfect clustering, entropy value is zero, whereas the worst possible entropy value is log2m. [13]

2.4 Kmeans clustering

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are: [14]

- The centroids of the K clusters, which can be used to label new data
- Labels for the training data (each data point is assigned to a single cluster)

Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have formed organically. The "Choosing K" section below describes how the number of groups can be determined.

Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents. [14]

$$E = \sum_{i=1}^{c} \sum_{x \in C_i} d(x, m_i) \qquad (1)$$

In the above equation, m_i is the center of cluster C_i, while d(x, m_i) is the Euclidean distance between a point x and m_i. Thus, the criterion function *E* attempts to minimize the distance of each point from the center of the cluster to which the point belongs. More specifically, the algorithm begins by initializing a set of *c* cluster centers. Then, it assigns each object of the dataset to the cluster whose center is the nearest, and recomputes the centers. The process continues until the centers of the clusters stop changing.

The K-means clustering algorithm uses iterative refinement to produce a final result. The algorithm inputs are the number of clusters K and the data set. The data set is a collection of features for each data point. The algorithms starts with initial estimates for the K centroids, which can either be randomly generated or randomly selected from the data set. The algorithm then iterates between two steps: [14]

1. Data assigment step:

Each centroid defines one of the clusters. In this step, each data point is assigned to its nearest centroid, based on the squared Euclidean distance. More formally, if ci is the collection of centroids in set C, then each data point ***x*** is assigned to a cluster based on: [14]

$$\underset{c_i \in C}{argmin}\, dist(c_i, x)^2 \quad \textbf{(2)}$$

Where *dist*( ) is the standard (L_2) Euclidean distance. Let the set of data point assignments for each i^{th} cluster centroid be S_i.

2. Centroid update step:

In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid's cluster. [13]

$$C_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i \quad \textbf{(3)}$$

The algorithm iterates between steps one and two until a stopping criteria is met (i.e., no data points change clusters, the sum of the distances is minimized, or some maximum number of iterations is reached).

Choosing K

The Kmeans algorithm finds the clusters and data set labels for a particular pre-chosen K. To find the number of clusters in the data, the user needs to run the K-means clustering algorithm for a range of K values and compare the results. In general, there is no method for determining exact value of K, but an accurate estimate can be obtained using the following techniques.

One of the metrics that is commonly used to compare results across different values of K is the mean distance between data points and their cluster centroid. Since increasing the number of clusters will always reduce the distance to data points, increasing K will always decrease this metric, to the extreme of reaching zero when K is the same as the number of data points. Thus, this metric cannot be used as the sole target. Instead, mean distance to the centroid as a function of K is plotted and the "elbow point," where the rate of decrease sharply shifts, can be used to roughly determine K. [14]

3 BICLUSTERING

3.1 Definition

Biclusters are represented in the literature in different ways, where genes can be found in either rows or columns, and different names refer the same expression sub-matrix.

In biclustering homogeneous subgroups, or biclusters, do not necessarily span all the columns. This makes biclustering useful for identifying possible relevant subspaces in the data.

Now on, $\mathscr{B}$ be a bicluster consisting of a set I of **/I/** genes and a set J of **/J/** conditions, in which refers to the expression level of gene i under sample j. Then $\mathscr{B}$ can be represented as follows: [15]

$$\mathscr{B} = \begin{pmatrix} a_{11}a_{12} \cdots a_{1|J|} \\ a_{21}\ a_{22} \cdots a_{2|J|} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ a_{|I|1}\ a_{|I|2} \cdots a_{|I||J|} \end{pmatrix}$$

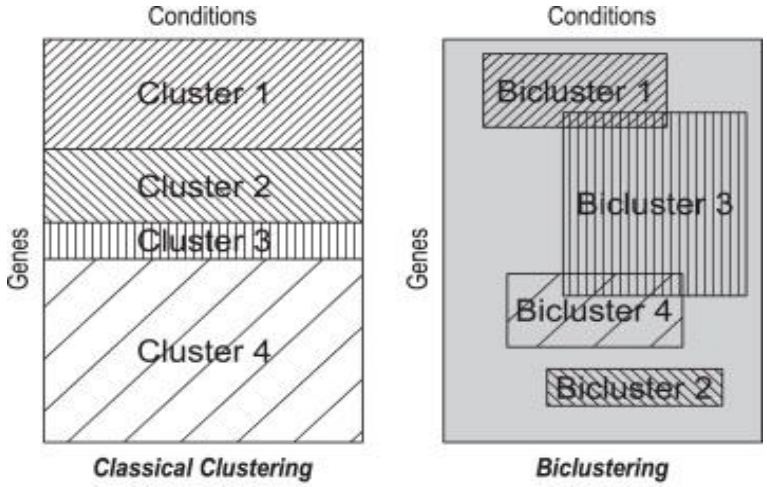
Where the gene g_i is the i^{th} row, i.e. $g_i = \{a_{i1}, a_{i2} \ldots ai_{|J|}\}$, and condition c_j is the j^{th} column, *i.e.* $c_j = \{a_{1j}, a_{2j}, \ldots a_{|I|j}\}$,

Genes and conditions means in biclusters are frequently used in several evaluation measure definitions. We represent these values as a_{iJ} and a_{Ij} referring to the i row (gene) and j column (conditions) means, respectively. Furthermore, the mean of all the expression values in $\mathscr{B}$ is referred to as b_{IJ}. [15]

	Condition 1	…	Condition *j*	…	Condition *m*
Gene 1	a_{11}	...	a_{1j}	...	a_{1m}
Gene ...	...	...	...	...	...
Gene *i*	a_{i1}	...	a_{ij}	...	a_{im}
Gene …	...	...	...	...	...
Gene *n*	a_{n1}	...	a_{nj}	...	a_{nm}

➢ **Table 2. 1 microarray matrix**

Such a matrix A, with n rows and m columns, is defined by its set of rows, $X=\{x_1,\ldots,x_n\}$, and its set of columns, $Y=\{y_1,\ldots,y_n\}$. We will use (X,Y) to denote the matrix A. If $I \subseteq X$ and $J \subseteq Y$, are subsets of the rows and columns, respectively, $A_{IJ} = (I,J)$ denotes the sub-matrix A_{IJ} of A that contains only the elements a_{ij} belonging to the sub-matrix with set of rows I and set of columns J.



➢ **Figure 2. 5 Biclustering vs Clustering**

3.2 Bicluster Types

An interesting criteria to evaluate a biclustering algorithm concerns the identification of the type of biclusters the algorithm is able to find. We identified four major classes of biclusters: [15]

(a) Biclusters with constant values.

(b) Biclusters with constant values on rows or columns.

(c) Biclusters with coherent values.

(d) Biclusters with coherent evolutions.

The simplest biclustering algorithms identify subsets of rows and subsets of columns with constant values. An example of a constant bicluster is presented in Figure 2.6(a). These algorithms are studied in section 3.2(a).

Other biclustering approaches look for subsets of rows and subsets of columns with constant values on the rows or on the columns of the data matrix. The bicluster presented in Figure 2.6(b) is an example of a bicluster with constant rows, while the bicluster depicted in Figure2.6(c) is an example of a bicluster with constant columns. section 3.2(b) studies algorithms that discover biclusters with constant values on rows or columns. [15]

More sophisticated biclustering approaches look for biclusters with coherent values on both rows and columns. The biclusters in Figure 2.6(d) and Figure 2.6(e) are examples of this type of bicluster, where each row and column can be obtained by adding a constant to each of the others or by multiplying each of the others by a constant value. These algorithms are studied in section 3.2(c).

The last type of biclustering approaches we analyzed addresses the problem of finding biclusters with coherent evolutions. These approaches view the elements of the matrix as symbolic values, and try to discover subsets of rows and subsets of columns with coherent behaviors regardless of the exact numeric values in the data matrix. The co-evolution property can be observed on the entire bicluster, that is on both rows and columns of the sub-matrix (see Figure .2.6(f)), on the rows of the bicluster (see Figure 2.6(g)), or on the columns of the bicluster (see Figure 2.6(h) and Figure 2.6(i)). These approaches are addressed in section 3.2(d) [15]

1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0

(a) Constant bicluster

1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0
4.0	4.0	4.0	4.0

(b) Constant rows

1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0

(c) Constant columns

1.0	2.0	5.0	0.0
2.0	3.0	6.0	1.0
4.0	5.0	8.0	3.0
5.0	6.0	9.0	4.0

(d) Coherent values additive model

1.0	2.0	0.5	1.5
2.0	4.0	1.0	3.0
4.0	8.0	2.0	6.0
3.0	6.0	1.5	4.5

(e) Coherent values multiplicative model

S1	S1	S1	S1
S1	S1	S1	S1
S1	S1	S1	S1
S1	S1	S1	S1

(f) Overall coherent evolution

S1	S1	S1	S1
S2	S2	S2	S2
S3	S3	S3	S3
S4	S4	S4	S4

(g) Coherent evolution on the rows

S1	S2	S3	S4
S1	S2	S3	S4
S1	S2	S3	S4
S1	S2	S3	S4

(h) Coherent evolution on the columns

70	13	19	10
29	40	49	35
40	20	27	15
90	15	20	12

(i) Coherent evolution on the columns

➢ **Figure 2. 6 Examples of Different Types of Biclusters**

Given the data matrix ***A=(X,Y)***, with set of rows ***X*** and set of columns ***Y***, a bicluster is a sub-matrix ***(I,J)***, where ***I*** is a subset of the rows ***X, J*** is a subset of the columns ***Y*** and a_{ij} is the value in the data matrix ***A*** corresponding to row ***i*** and column ***j***. We denote by a_{iJ} the mean of the i^{th} row in the bicluster, a_{Ij} the mean of the j^{th} column in the bicluster and a_{ij} the mean of all elements in the bicluster. These values are defined by: [15]

$$a_{ij} = \frac{1}{|J|}\sum_{j\in J} a_{ij} \quad \textbf{(4)}$$

$$a_{Ij} = \frac{1}{|I|}\sum_{i\in I} a_{ij} \quad \textbf{(5)}$$

$$a_{IJ} = \frac{1}{|I||J|}\sum_{i\in I, j\in J} a_{ij} = \frac{1}{|I|}\sum_{i\in I} a_{iJ} = \frac{1}{|J|}\sum_{j\in J} a_{Ij} \quad \textbf{(6)}$$

(a) Biclusters with Constant Values

When the goal of a biclustering algorithm is to find a constant bicluster or several constant biclusters, it is natural to consider ways of reordering the rows and columns of the data matrix in order to group together similar rows and similar columns, and discover subsets of rows and subsets of columns (biclusters) with similar values. Since this approach only produces good results when it is performed on non-noisy data, which does not correspond to the great majority of available data, more sophisticated approaches can be used to pursue the goal of finding biclusters with constant values. When gene expression data is used, constant biclusters reveal subsets of genes with similar expression values within a subset of conditions. The bicluster in Figure 2.6(a) is an example of a bicluster with constant values.

A perfect constant bicluster is a sub-matrix ***(I,J)***, where all values within the bicluster are equal for all $i \in I$ and all $j \in J$: [21]

$$a_{ij} = \mu \quad \textbf{(7)}$$

Although these "ideal" biclusters can be found in some data matrices, in real data, constant biclusters are usually masked by noise. This means that the values aij found in what can be considered a constant bicluster are generally presented as $a_{ij} + \mu$ where n_{ij} is the noise associated

with the real value $\boldsymbol{\mu}$ of $\boldsymbol{a_{ij}}$. The merit function used to compute and evaluate constant biclusters is, in general, the variance or some metric based on it.

(b) Biclusters with Constant Values on Rows or Columns

There exists great practical interest in discovering biclusters that exhibit coherent variations on the rows or on the columns of the data matrix. As such, many biclustering algorithms aim at finding biclusters with constant values on the rows or the columns of the data matrix. The biclusters in Figure 2.6(b) and Figure 2.6(c) are examples of biclusters with constant rows and constant columns, respectively. In the case of gene expression data, a bicluster with constant values in the rows identifies a subset of genes with similar expression values across a subset of conditions, allowing the expression levels to differ from gene to gene. The same reasoning can be applied to identify a subset of conditions within which a subset of genes present similar expression values assuming that the expression values may differ from condition to condition. [15]

A perfect bicluster with constant rows is a sub-matrix $\boldsymbol{(I,J)}$, where all the values within the bicluster can be obtained using one of the following expressions:

$$a_{ij} = \mu + \alpha_i \quad \textbf{(8)}$$

$$a_{ij} = \mu \times \alpha_i \quad \textbf{(9)}$$

where $\boldsymbol{\mu}$ is the typical value within the bicluster and $\boldsymbol{\alpha_i}$ is the adjustment for row $\boldsymbol{i} \in \boldsymbol{I}$. This adjustment can be obtained in either an additive (8) or multiplicative way (9).

Similarly, a perfect bicluster with constant columns is a sub-matrix $\boldsymbol{(I,J)}$, where all the values within the bicluster can be obtained using one of the following expressions: [15]

$$a_{ij} = \mu + \beta_j \quad \textbf{(10)}$$

$$a_{ij} = \mu \times \beta_j \quad \textbf{(11)}$$

where $\boldsymbol{\mu}$ is the typical value within the bicluster and $\boldsymbol{\beta_j}$ is the adjustment for column $\boldsymbol{j} \in \boldsymbol{J}$.

This class of biclusters cannot be found simply by computing the variance of the values within the bicluster or by computing similarities between the rows and columns of the data matrix as we have seen in section 3.2(a).

(c) Biclusters with Coherent Values

An overall improvement over the methods considered in the previous section, which presented biclusters with constant values either on rows or columns, is to consider biclusters with coherent values on both rows and columns. In the case of gene expression data, we can be interested in identifying more complex biclusters where a subset of genes and a subset of conditions have coherent values on both rows and columns. The biclusters in Figure 2.6(d) and Figure 2.6(e) are examples of this type of biclusters. [15]

This class of biclusters cannot be found simply by considering that the values within the bicluster are given by additive or multiplicative models that consider an adjustment for either the rows or the columns, as it was described in (8), (9), (10) and (11). More sophisticated approaches perform an analysis of variance between groups and use a particular form of co-variance between both rows and columns in the bicluster to evaluate the quality of the resulting bicluster or set of biclusters.

Following the same reasoning of section 3.2(b), the biclustering algorithms that look for biclusters with coherent values can be viewed as based on an *additive model*. When an additive model is used within the biclustering framework, a perfect bicluster with coherent values, ***(I,J)***, is defined as a subset of rows and a subset of columns, whose values a_{ij} are predicted using the following expression: [15]

$$a_{ij} = \mu + \alpha_i + \beta_j \qquad \textbf{(12)}$$

where μ is the typical value within the bicluster, α_i is the adjustment for row $i \in I$ and β_j is the adjustment for column $j \in J$. The bicluster in Figure 2.6(d) is an example of a bicluster with coherent values on both rows and columns, whose values can be described using an additive model. The biclusters in Figure 2.6(b) and Figure 2.6(c) can be considered special cases of this general additive model where the coherence of values can be observed on the rows and on the columns of the bicluster, respectively. This means that (8) and (10) are special cases of the model represented by (12) when $\alpha_i = 0$ and $\beta_j = 0$, respectively.

(d) Biclusters with Coherent Evolutions

In the previous section we revised several biclustering algorithms that aimed at discovering biclusters with coherent values. Other biclustering algorithms address the problem of finding coherent evolutions across the rows and/or columns of the data matrix regardless of their exact values. In the case of gene expression data, we may be interested in looking for evidence that a subset of genes is up-regulated or down-regulated across a subset of conditions without taking into account their actual expression values in the data matrix. The co-evolution property can be observed on both rows and columns of the biclusters, as it is shown in Figure 2.6(f), on the rows of the bicluster or on its columns. The biclusters presented in Figure 2.6(h) and Figure 2.6(i) are examples of biclusters with coherent evolutions on the columns, while Figure 2.6(g) shows a bicluster with co-evolution on the rows. [15]

A bicluster is a group of rows whose values induce a linear order across a subset of the columns. Their work focus on the relative order of the columns in the bicluster rather than on the uniformity of the actual values in the data matrix as the plaid model did. A sub-matrix is order-preserving if there is a permutation of its columns under which the sequence of values in every row is strictly increasing. The bicluster presented in Figure 2.6(i) is an example of an OPSM, where $ai4 \leq ai2 \leq ai3 \leq ai1$, and represents a bicluster with coherent evolution on its columns.

Furthermore, a complete model as the pair ***(J,π)***, where ***J*** is a set of s columns and $\pi = (j_1, j_2, j_s)$ is a linear ordering of the columns in ***J***. They say that a row supports ***(J,π)*** if the ***s*** corresponding values, ordered according to the permutation π are monotonically increasing. [15]

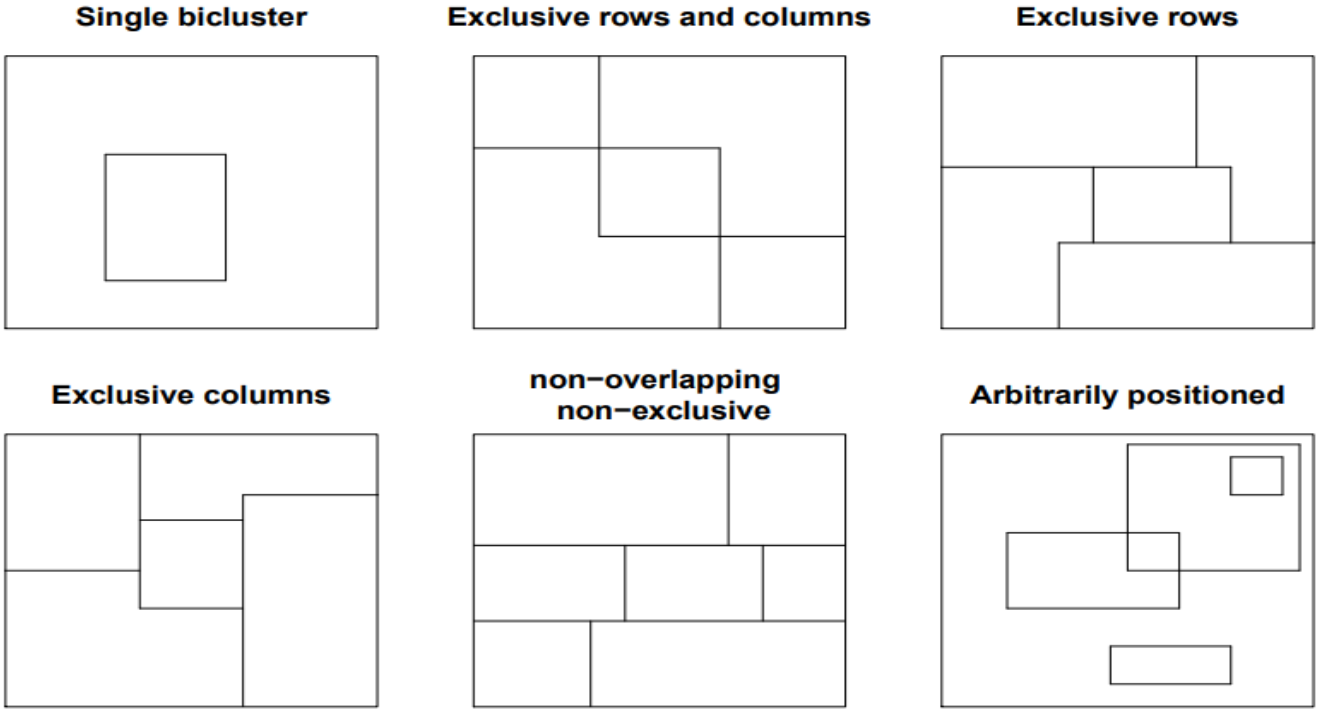
3.3 Bicluster structure

It is also interesting classify the biclustering methods regarding to the way in which rows and columns from the input matrix are incorporated in biclusters. We named this as Bicluster Structure. In this sense, we can define the follows structures: [16]

1) Single bicluster.
2) Exclusive row and column biclusters.
3) Exclusive-rows or exclusive-columns biclusters.
4) Non-overlapping non-exclusive biclusters.
5) Arbitrarily positioned overlapping biclusters.

Structure type1 is self-explanatory as the method is just able to find one single bicluster at a time. In type2 every bicluster consists of an exclusive subset of rows and columns. Although this can be the first approach to extract relevant knowledge from gene expression data, it has long been recognized that such an structure will seldom exist in real data. Genes are very likely to participate in more than just one biological function and therefore should be included in more than one bicluster. The same of course holds for the conditions. That is where it come to biclusters of type3. Improvements of this structures are the non-overlapping and non-exclusive biclusters, which exhaustively splits the matrix in biclusters which overlap in just one dimension. But still, this methods have some restrictions, such as every row and column in the data has to belong to at least one bicluster. However, in the context of gene expression it is very likely that some rows or columns do not belong to any bicluster at all. Thus, there are algorithms which are able to observe arbitrarily positioned overlapping biclusters. This sophisticated methods allow bicluster to be non-exclusive, non-exhaustive and overlapping in both dimensions. [16]

When it comes to the analysis of gene expression data one wants to be able to observe overlapping, non-exclusive and non-exhaustive biclusters with coherent values (either additive or multiplicative). The reasons is genes participate in more than one biological function and some conditions share certain gene expression alterations (non-exclusive and overlapping), some genes are not relevant at all (non-exhaustive) in an certain experiment and at last, gene expression level varies between different genes and samples and one does not want to lose information (coherent values), have already been mentioned above. [16]



➢ **Figure 2. 7 Examples of the different bicluster structures**

3.4 Systematic and stochastic biclustering algorithms

The biclustering problem is NP-hard, consequently, heuristic search algorithms are typically used to approximate the problem by finding sub-optimal solution.

We distinguish two main classes of biclustering algorithms: systematic search algorithms and stochastic search algorithms, also called metaheuristic algorithms. [17]

3.4.1 SYSTEMATIC BICLUSTERING ALGORITHMS

Systematic search algorithms are based on one of the following general approaches.

1. **Divide-And-Conquer (DAC)** approach: Generally, this approach divides repeatedly the problem into smaller subproblems with similar structures to the original problem, until these subproblems become smaller enough to be solved directly. The solutions to the subproblems are then combined to create a solution to the original problem. With this approach, we start by a bicluster representing the whole data matrix then we partition this matrix in two submatrices to obtain two biclusters. We reiterate recursively this process until we obtain a certain number of biclusters verifying a specific set of properties. For instance, *Prelic et al.* partition the data matrix *M'* (*M'* is a discretization of a data matrix *M* which contains only binary values where a cell m_{ij} contains 1 if $gene_i$ is expressed under $condition_j$ and 0 otherwise) into three submatrices, one of which contains only 0-cells. The algorithm is then recursively applied to the remaining two submatrices, and ends if the current matrix represents a bicluster which contains only 1's. The advantage of this approach is that it is fast, however, its biggest disadvantage is that it may ignore good biclusters by partitioning them before identifying them. [17]

2. **Greedy Iterative Search (GIS)** approach: This approach constructs a solution in a step-by-step way using a given quality criterion. Decisions made at each step are based on information at hand without worrying about the effect these decisions may have in the future. Moreover, once a decision is taken, it becomes irreversible and is never reconsidered. By applying this approach to the biclustering problem, at each iteration, we construct submatrices of the data matrix by adding/removing a row/column to/from the current submatrix that maximizes/minimizes a certain function. We reiterate this process until no other row/column can be added/removed to/from any submatrix. For instance, the algorithm *Maximum Similarity Biclusters* starts by constructing a similarity matrix based on a reference gene. A greedy strategy of removing rows/columns iteratively is employed to provide the maximum similarity bicluster in polynomial time. *Shabalin et al*. extract large average submatrices according to a *Bonferroni-based* significance score. Several graph-theoretical approaches have also been proposed. Another recent work was reported by *Ayadi et al*. which constructs a *Directed Acyclic Graph* (DAG) to combine a subset of genes under a subset of conditions iteratively, by adopting the evaluation functions E_{ACSI} and E_{ASR}. [17]

3. **Biclusters Enumeration (BE)** approach: This approach tries to enumerate (explicitly or implicitly) all the solutions for an original problem. The enumeration process is generally represented by a search tree. By applying this approach to the biclustering problem, we identify all the possible groups of biclusters in order to keep the best one. *Ayadi et al.* use a Bicluster Enumeration Tree (BET) to find all the biclusters (nodes) of interest, reachable from the root of the BET, by adopting to the E_{ASR} evaluation function.

To reduce the size of the BET, a quality threshold is employed to cut branches that cannot lead to biclusters of desired quality. This approach has the advantage of being able to obtain the best solutions. Its disadvantage is that it is costly in computing time and in memory space. [17]

3.4.2 STOCHASTIC BICLUSTERING ALGORITHMS

Stochastic search algorithm are based on one of the following general approaches.

1. **Neighborhood Search (NS)** approach: Neighborhood search, also called local search, is based on the notion of neighborhood and a strategy exploiting this neighborhood. A neighborhood search algorithm starts with an initial solution s and then moves iteratively to a neighboring solution thanks to the neighborhood exploitation strategy. A neighboring solution is generally generated by applying a transformation operator, also called move operator, to the current solution. At each iteration, the neighborhood exploitation strategy decides the neighboring solution.
   By applying this approach to the biclustering problem, we start by an initial solution which can be a cluster, a bicluster or the whole matrix. Then, at each iteration, we try to improve this solution by adding and/or removing some genes/conditions to minimize/maximize a certain function. The difference with the greedy search algorithms is that if we delete, for example, one gene/condition, we can later add this gene/condition to the solution.
   Cheng and Church are probably the first to apply this concept to the biclustering problem. Their goal is to find biclusters with a E_{MSR} value lower than a fixed threshold. Hence, they proposed a local search procedure which deletes/adds genes/conditions to the biclusters. The multiple node deletion method removes all genes and conditions with a E_{MSR} score lower than a fixed threshold. The single node deletion method iteratively removes the gene or column that has low quality according to E_{MSR}. Finally, the node addition method adds genes and conditions that do not decrease the quality of the actual bicluster. In order to find a given number of biclusters, this approach is iteratively executed on the remaining genes and conditions that are not present in the previous obtained biclusters.
   The move operator used by *Ayadi et al*. is based on the drop/add operation which removes a $gene_i$ where $i \in I'$ from the bicluster $b=(I',J')$ and adds one $gene_v$, where $v \notin$ I', or various $gene_v,\ldots,gene_w$, where $v \notin I',\ldots,w \notin I'$, to b. The move operator can be defined as follows: we first choose a pair of genes $\{gene_i, gene_j\}$ from b which have a bad quality according to an evaluation function. Such a pair of genes contribute negatively to the quality of the bicluster b. Then we look for other pairs of genes $\{gene_j, gene_{r1}\}$, $\{gene_j, gene_{r2}\},\ldots,\{gene_j, gene_n\}$, where $r_1 \notin I',\ldots, r_n \notin I'$ which have a good quality according to the evaluation function. Hence, $gene_j$ contributes positively to the quality of the bicluster when it is associated with $gene_{r1}\ \ldots gene_{rn}$. Finally, we replace $gene_i$ in b by $gene_{r1}, gene_{r2}\ldots gene_{rn}$.
   The advantage of this approach lies in the ability to explore large search spaces. This approach also offers the possibility of trade-off between solution quality and running time. Indeed, when the quality of a solution tends to improve gradually over time, the user can stop the execution at a chosen time. The disadvantage of this approach is that the search lead to sub-optimal solutions (local maxima). [17]

2. **Evolutionary Computation (EC)** approach:
   The evolutionary computation approach is based on the natural evolutionary process such as population, reproduction, mutation, recombination, and selection. Candidate solutions of the given problem are sampled by a set of individuals in a population. An evaluation mechanism (fitness evaluation) is established to assess the quality of each individual. Evolution operators eliminate some (less fit) individuals and produce new individuals from selected individuals.
   By applying this approach to the biclustering problem, we start from an initial population of solutions, *i.e.*, clusters, biclusters or the whole matrix, then, we measure the quality of each solution of the population by the fitness function. We select a number of solutions to produce new solutions by recombination and mutation operators. This process ends when a prefixed stop condition is verified. For instance, *Divina and Aguilar-Ruiz* generate a population representing biclusters of dimension one because these biclusters have a high E_{MSR} score. From this population, selection, crossover and mutation are repeatedly applied to the population. A number of biclusters are selected for reproduction with a tournament selection operator. In other words, a certain number of biclusters are first selected randomly, and the best one according to E_{MSR} is chosen. Each selected pair of parents is recombined by a crossover operator. For this, three crossover (one point, two points and uniform) operators are applied, with equal probability. The resulting offspring is mutated by using three mutation operators: the standard mutation operator, a mutation operator that adds a row and a column to the bicluster. Since mutation is a highly random operation, it is applied with a low probability. The process is repeated with the new generation of offspring, until a maximum number of generations is reached.
   This approach shares the similar advantages and disadvantages with the neighborhood search approach. [17]

3. **Hybrid (H)** approach: The hybrid approach, also called memetic approach, tries to combine both the neighborhood search and the evolutionary approaches. This hybrid approach is known to be quite successful in solving many hard combinatorial search problems. The purpose of such an approach is to take advantage of the complementary nature of the evolutionary and neighborhood search methods. Indeed, it is generally believed that the evolutionary framework offers more facilities for exploration, while neighborhood search has more capability for exploitation. Combining them may offer a better balance between exploitation and exploration which is highly desirable for an effective search.
   Mitra and Banka present a *Multi-Objective Evolutionary Algorithm (MOEA)* based on Pareto dominancy. The authors try to find biclusters with maximum size and homogeneity by using a multi-objective genetic algorithm called *Non-dominated Sorting Genetic Algorithm (NSGA-II)* in combination with the local search procedure. *Gallo et al.* present another hybrid algorithm based on *MOEA* combined with a local search strategy. They extract biclusters with multiple criteria like maximum rows, columns, homogeneity and row variance. A mechanism for re-orienting the search in terms of row variance and size is provided. The mutation operator is performed when the individual needs to be mutated by means of the probability assigned to the operator. Hence, the gene/condition of the bicluster is mutated at a random position. The crossover operator is applied over both the genes and the conditions. Hence, when both children are obtained by combining at the end and at the center each of the two parents, the individual to select as the only descendant is the non-dominated one. [16]

3.5 Bicluster Algorithms

Bimax

Bimax is an algorithm due to Prelic et al. It uses a simple data model reflecting the fundamental idea of biclustering, while aiming to determine all optimal biclusters in reasonable time. This method has the benefit of providing a basis to investigate the usefulness of the biclustering concept in general, independently of interfering effects caused by approximate algorithms, and the effectiveness of more complex scoring schemes and biclustering methods in comparison to a plain approach. [18]

Binary datasets represent a compact and simple way to store data about the relationships between a group of objects and their possible properties. This type of data is present in many research fields, including data mining, text mining, bioinformatics, engineering or paleontology, among others. What the values 0 and 1 stand for depends on the context. For example, when working with gene and conditions features, if gene r combined with a condition c, then (r, c) is equal to 1; otherwise, it is equal to 0. Commonly, the binary values 1 and 0 mean that under experimental condition c, gene r is either expressed or not, respectively.

Prelic et al. (2006) used this method to compare different other algorithms to a constant benchmark, but the method is also useful in many other application fields where binary or quasi-binary data is used. [18]

```
1.    procedure Bimax(E)
2.      Z ← Ø
3.      M ← conquer(E,({1,...,n},{1,...,m}),Z)
4.      return M
5.    end procedure

6.    procedure conquer(E,(G,C),Z)
7.      if ∀i ∈ G,j ∈ C: e_ij = 1 then
8.        return {(G,C)}
9.      end if
10.     (Gu,Gv,Gw,Cu,Cv) = divide(E,(G,C),Z)
11.     Mu ← Ø, Mv ← Ø
12.     if Gu ≠ Ø then
13.       Mu ← conquer(E,(Gu ∪ Gw,Cu),Z)
14.     end if
15.     if Gv ≠ Ø AND Gw ≠ Ø then
16.       Mv ← conquer(E,(Gv,Cv),Z)
17.       else if Gw ≠ Ø then
18.       Z' ← Z ∪ {Cv}
19.       Mv ← conquer(E,(Gw ∪ Gv, Cu ∪ Cv),Z)
20.     end if
21.     return Mu ∪ Mv
22.   end procedure

23.   procedure divide(E,(G,C),Z)
24.     G' ← reduce(E,(G,C),Z)
25.   chose i ∈ G' with 0 < ∑_{j∈C} e_ij </C/
26.   if such an i ∈ G' exist then
```

```
27.        Cu ← {j | j ∈ C AND e_ij = 1}
28.      else
29.        Cu = C
30.      end if
31.      Cv ← C\Cu
32.      Gu ← Ø, Gv ← Ø, Gw ← Ø
33.      for each i ∈ G' do
34.        C* ← {j | j ∈ C and e_ij = 1}
35.      if C* ⊆ Cu then
36.        Gu ← Gu ∪ {i}
37.      else if C* ⊆ Cu then
38.        Gv ← Gv ∪ {i}
39.      else
40.        Gw ← Gw ∪ {i}
41.      end if
42.      end for
43.      return (Gu,Gv,Gw,Cu,Cv)
44.      end procedure


45.      procedure reduce(E,(G,C),Z)
46.        G' ← Ø
47.        for each i ∈ G do
48.          C* ← {j | j ∈ C and eij = 1}
49.          if C* ≠ Ø AND ∀ C+ ∈ Z: C+ ∩ C* ≠ Ø then
50.            G' = G' ∪ {i}
51.          end if
52.        end for
53.        return G'
54.      end procedure
```

➢ **Figure 2. 8 The Bimax algorithm**

The model assumes two possible expression levels per gene: no change and change with respect to a control experiment (To this end, a preprocessing step normalizes log expression values and then transforms matrix cells into discrete values, e.g. by using a 2-fold change cutoff.).

ISA

The iterative signature algorithm of *Bergmann et al. (2003)* for bicluster contains very high or very low values. It starts with a random set of rows and iterates between normalized rows and normalized columns to find the largest subgroup of extreme values. Due to the normalization quantiles of the normal distributions can be used to identify extreme values. In each iteration the corresponding row or column vector is updated until changes no longer occur. [18]

The algorithm, presented in Figure 2.9, uses two normalized copies of the original gene expression matrix. The matrix E^G has rows normalized to mean 0 and variance 1 and the matrix E^C has columns normalized similarly. We denote by $e^G_{uV'}$ the mean expression of genes from *V'* in the sample *u* and by $e^C_{U'v}$ v the mean expression of the gene *v* in samples from *U'*. A *bicluster B = (U',V')* is required, Here T_G is the threshold parameter and σ_G is the standard deviation of the means $e^G_{U'v}$ where *v* ranges over all possible genes and *U'* is fixed. Similarly, T_C, σ_C are the corresponding parameters for the column set *V'*. The idea is that if the genes in *V'* are up- or down-

regulated in the conditions U' then their average expression should be significantly far (*i.e.*, T_G standard deviations) from its expected value on random matrices (which is 0 since the matrix is standardized). A similar argument holds for the conditions in U'. The standard deviations can be predicted as $\frac{1}{\sqrt{|U'|}}, \frac{1}{\sqrt{|V'|}}$ being a linear sum of $|U'|$ (or$|V'|$) independent standard random variables. Alternatively, the standard deviations can be estimated directly from the data, correcting for possible biases in the statistics of the specific condition and gene sets used. In other words, in a bicluster, the *z*-score of each gene, measured. The biclusters samples, and the *z*-score of each sample, measured the biclusters samples, should exceed a threshold. As we shall see below, ISA will not discover biclusters for which the conditions hold strictly, but will use a relaxed version.

The algorithm starts from an arbitrary set of genes $V' = V_{in}$. The set may be randomly generated or selected based on some prior knowledge. The algorithm then repeatedly applies the update equations: [18]

$$U_i = \{u \in U : |e^C_{uV_i}| > T_C \sigma_C\}, V_{i+1} = \{v \in V : |e^G_{U_i v}| > T_G \sigma_G\}$$

The iterations are terminated at step *n* satisfying: [18]

$$\frac{|V_{n-1} \setminus V_{n-i-1}|}{|V_{n-i} \cup V_{n-i-1}|} < \varepsilon$$

For all *i* smaller than some *m*. The ISA thus converges to an approximated fixed point that is considered to be a bicluster. The actual fixed point depends on both the initial set V_{in} and the threshold parameters T_C,T_G. To generate a representative set of biclusters, it is possible to run ISA with many different initial conditions, including known sets of associated genes or random sets, and to vary the thresholds. After eliminating redundancies (fixed points that were encountered several times), the set of fixed points can be analyzed as a set of biclusters.

ISA($U, V, E, V_{in}, T_G, T_C, m, \epsilon$):
U : conditions. V : genes.
E : Gene expression matrix.
V_{in} : Initial gene set.
T_G, T_C: gene and condition z-score thresholds.
m, ϵ: stopping criteria.
Construct a column standardized matrix E^C.
Construct a row standardized matrix E^G.
Initialize counters $n = 0, n' = 0$.
Initialize the current genes set $V' = V_{in}$
Initialize an empty condition set U'.
While $(n - n' < m)$ do
    Compute $e^C_{uV'} = \frac{1}{|V'|} \sum_{v \in V'} e^C_{uv}$ for $u \in U$.
    $U' = \{u \in U : |e^C_{uV'}| > \frac{T_C}{\sqrt{|V'|}}\}$
    Compute $e^G_{U'v} = \frac{1}{|U'|} \sum_{u \in U'} e^G_{uv}$ for $v \in V$.
    $V'' = V'$
    $V' = \{v \in V : |e^G_{U'v}| > \frac{T_G}{\sqrt{U'}}\}$
    if $(\frac{|V' \setminus V''|}{|V' \cup V''|} < \epsilon)$ then $n' = n$
    $n = n + 1$
Report U', V'

➢ **Figure 2. 9 The ISA algorithm**

Plaid Models

The original plaid models for biclustering, defined by *Lazzeroni and Owen (2002)*, fit layers k to the model.

$$aij = (\mu_0 + \alpha_{i0} + \beta_{j0}) + \sum_{k=1}^{k}(\mu_k + \alpha_{ik} + \beta_{jk})\rho_{ik}k_{jk} + \varepsilon_{ij} \qquad \textbf{(13)}$$

using ordinary least squares (OLS), where ***μ, α, β*** represent mean, row and column effects and ***ρ*** and ***κ*** identify if a row or column is member of the layer, respectively. *Turner et al. (2005)* replaced the *OLS* with a binary least square algorithm and obtained better results. [18]

Plaid(U, V, E, S):
U : conditions. V : genes.
E : Gene expression matrix.
S: maximum cycles per iteration.
Set $K = 0$
adding a new layer:
K=K+1
Compute initial values of $\kappa_{jK}^{(0)}, \rho_{iK}^{(0)}$. Set $s = 1$
While $(s \leq S)$ do:
Compute $\mu_K^{(s)}, \alpha_{iK}^{(s)}, \beta_{jK}^{(s)}$ using equations (7)- (9).
Compute $\kappa_K^{(s)}$ using equations (11)
Compute $\rho_K^{(s)}$ using equations (10)
If $\rho_K^{(s)} > 0.5$ set $\rho_K^{(s)} = 0.5 + s/2S$, else set $\rho_K^{(s)} = 0.5 - s/2S$
If $\kappa_K^{(s)} > 0.5$ set $\kappa_K^{(s)} = 0.5 + s/2S$, else set $\kappa_K^{(s)} = 0.5 - s/2S$
If the importance of layer K is non random then record the layer and repeat
Else exit.
Report layers $1, \ldots, K-1$.

➢ **Figure 2. 10 The Plaid model algorithm**

Spectral Biclustering

The bicluster algorithm described by *Kluger et al. (2003)* uses a singular value decomposition and the resulting eigenvalues and eigenvectors to retrieve bicluster from the data. This leads to a checkerboard bicluster structure. The algorithm is very sensitive to data variations and therefore needs a very careful preprocessing which *Kluger et al. (2003)* included into their algorithm. A chosen upper border for the variance within the bicluster determines the number of bicluster. [18]

Spectral biclustering approaches use techniques from linear algebra to identify bicluster structures in the input data. Here we review the biclustering technique presented in *Kluger et al*. In this model, it is assumed that the expression matrix has a hidden checkerboard-like structure that we try to identify using eigenvector computations. The structure assumption is argued to hold for clinical data, where tissues cluster to cancer types and genes cluster to groups, each distinguishing a particular tissue type from the other types. [18]

To describe the algorithm, suppose at first that the matrix *E* has a checkerboard-like structure (see Figure 2. 11). Obviously we could discover it directly, but we could also infer it using a technique from linear algebra that will be useful in case the structure is hidden due to row and column shufflings. The technique is based on a relation between the block structure of E and the block structure of pairs of eigenvectors for $E\,E^T$ and $E^T\,E$, which we describe next. First, observe that the eigenvalues of $E\,E^T$ and $E^T\,E$ are the same. Now, consider a vector *x* that is stepwise, *i.e.*, piecewise constant, and whose block structure matches that of the rows of *E*. Applying *E* to *x* we get a stepwise vector *y*. If we now apply E^T to *y* we get a vector with the same block structure as *x*. The same relation is observed when applying first E^T and then *E*. Hence, vectors of the stepwise pattern of *x* form a subspace that is closed under $E^T\,E$. This subspace is spanned by eigenvectors of this matrix. Similarly, eigenvectors of $E\,E^T$ span the subspace formed by vectors of the form of *y*. More importantly, taking now *x* to be an eigenvector of $E^T\,E$ with an eigenvalue λ, we observe that $y = Ex$ is an eigenvector of $E\,E^T$ with the same eigenvalue. [18]

$$Ex = \begin{bmatrix} 8 & 8 & 7 & 7 & 3 & 3 \\ 8 & 8 & 7 & 7 & 3 & 3 \\ 6 & 6 & 4 & 4 & 5 & 5 \\ 6 & 6 & 4 & 4 & 5 & 5 \end{bmatrix} \begin{bmatrix} a \\ a \\ b \\ b \\ c \\ c \end{bmatrix} = \begin{bmatrix} d \\ d \\ e \\ e \end{bmatrix} = y, E^T y = \begin{bmatrix} 8 & 8 & 6 & 6 \\ 8 & 8 & 6 & 6 \\ 7 & 7 & 4 & 4 \\ 7 & 7 & 4 & 4 \\ 3 & 3 & 5 & 5 \\ 3 & 3 & 5 & 5 \end{bmatrix} \begin{bmatrix} d \\ d \\ e \\ e \end{bmatrix} = \begin{bmatrix} a' \\ a' \\ b' \\ b' \\ c' \\ c' \end{bmatrix} = x' \qquad \textbf{(14)}$$

➢ **Figure 2. 11 An example of a checkerboard-like matrix**

This example of a checkerboard-like matrix E and the eigenvectors of E ET and ET E. The vector x satisfies the relation ET Ex = ET y = x' = λx. Similarly, y satisfies the equation E ET y = Ex = λy

In conclusion, the checkerboard-like structure of *E* is reflected in the stepwise structures of pairs of $E\,E^T$ and $E^T\,E$ eigenvectors that correspond to the same eigenvalue. One can find these eigenvector pairs by computing a singular value decomposition of *E*. Singular value decomposition is a standard algebraic technique, that expresses a real matrix *E* as a product $E = A\Delta B^T$, where Δ is a diagonal matrix and *A* and *B* are orthonormal matrices. The columns of *A* and *B* are the eigenvectors of $E\,E^T$ and $E^T\,E$, respectively. The entries of Δ are square roots of the corresponding eigenvalues, sorted in a non-increasing order. Hence, the eigenvector pairs are obtained by taking for each *i* the *i*th columns of *A* and *B*, and the corresponding eigenvalue is the Δ_{ii}^2. [18]

Spectral(U, V, E):
U : conditions. V : genes.
$E_{n\times m}$: Gene expression matrix.
Compute $R = diag(E \cdot 1_m)$ and $C = diag(1_n^T \cdot E)$.
Compute a singular value decomposition of $R^{-1/2}EC^{-1/2}$.
Discard the pair of eigenvectors corresponding to the largest eigenvalue.
For each pair of eigenvectors u, v of $R^{-1}EC^{-1}E^T$ and $C^{-1}E^TR^{-1}E$ with the same eigenvalue do:
    Apply k-means to check the fit of u and v to stepwise vectors.
Report the block structure of the p u, v with the best stepwise fit.

➢ **Figure 2. 12 The spectral biclustering**

For any eigenvector pair, one can check whether each of the vectors can be approximated using a piecewise constant vector. *Kluger et al*. use a one-dimensional k-means algorithm to test this fit. The block structures of the eigenvectors indicate the block structures of the rows and columns of *E*.

In the general case, the rows and columns of E are ordered arbitrarily, and the checkerboard like structure, if E has one, is hidden. To reveal such structure one computes the singular value decomposition of E and analyzes the eigenvectors of $E\,E^T$ and $E^T\,E$. A hidden checkboard structure will manifest itself by the existence of a pair of eigenvectors (one for each matrix) with the same eigenvalue, that are approximately piecewise constant. One can determine if this is the case by sorting the vectors or by clustering their values, as done in. [18]

SAMBA

The SAMBA algorithm (Statistical-Algorithmic Method for Bicluster Analysis) uses probabilistic modeling of the data and graph theoretic techniques to identify subsets of genes that *jointly respond* across a subset of conditions, where a gene is termed responding in some condition if its expression level changes significantly at that condition its normal level. Within the SAMBA framework, the expression data are modeled as a bipartite graph whose two parts correspond to conditions and genes, respectively, with edges for significant expression changes. The vertex pairs in the graph are assigned weights according to a probabilistic model, so that heavy subgraphs correspond to biclusters with high likelihood. Discovering the most significant biclusters in the data reduces under this weighting scheme to finding the heaviest subgraphs in the model bipartite graph. SAMBA employs a practical heuristic to search for heavy subgraphs. The search algorithm is motivated by a combinatorial algorithm for finding heavy bicliques that is exponential in the maximum gene degree in the graph. [18]

SAMBA($U, V, E, w, d, N_1, N_2, k$):
U : conditions. V : genes.
E : graph edges. w : edge/non-edge weights.
N_1, N_2 : condition set hashed set size limits. k : max biclusters per gene/condition.
Initialize a hash table $weight$
For all $v \in V$ with $|N(v)| \leq d$ do
  For all $S \subseteq N(v)$ with $N_1 \leq |S| \leq N_2$ do
    $weight[S] \leftarrow weight[S] + w(S, \{v\})$
For each $v \in V$ set $best[v][1 \ldots k]$ to the k heaviest sets S such that $v \in S$
For each $v \in V$ and each of the k sets $S = best[v][i]$
  $V' \leftarrow \cap_{u \in S} N(u)$.
  $B \leftarrow S \cup V'$.
  Do {
    $a = argmax_{x \in V \cup U}(w(B \cup x))$
    $b = argmax_{x \in B}(w(B \setminus x))$
    If $w(B \cup a) > w(B \setminus b)$ then $B = B \cup a$ else $B = B \setminus b$
  } **while** improving
  Store B.
Post process to filter overlapping biclusters.

- **Figure 2. 13 The SAMBA biclustering algorithm**

The SAMBA algorithm is based on representing the input expression data as a bipartite graph $G = (U,V,E)$. In this graph, U is the set of conditions, V is the set of genes, and $(u,v) \in E$ if v responds in condition u, that is, if the expression level of v changes significantly in u. A bicluster

corresponds to a subgraph $H = (U',V',E')$ of G, and represents a subset V' of genes that are co-regulated under a subset of conditions U'. The weight of a subgraph (or bicluster) is the sum of the weights of gene-condition pairs in it, including edges and non-edges. [18]

Coupled with the graph representation is a likelihood ratio model for the data. Let $H=(U',V',E')$ be a subgraph of G and denote $\overline{E'} = (U' \times V')\backslash E'$. For a vertex $w \in U' \cup V'$ let d_w denote its degree in G. The null model assumes that the occurrence of each edge (u,v) is an independent Bernoulli variable with parameter $p_{u,v}$. The probability $p_{u,v}$ is the fraction of bipartite graphs with degree sequence identical to G that contain the edge (u,v). In practice, one estimates $p_{u,v}$ using a Monte-Carlo process. This model tries to capture the characteristics of the different genes and conditions in the data. [18]

The alternative model assumes that each edge of a bicluster occurs with constant, high probability p_c. This model reflects the belief that biclusters represent approximately uniform relations between their elements. The log likelihood ratio for H is therefore: [18]

$$log\ L(H) = \sum_{(u,v)\in E'} log\frac{p_c}{p_{u,v}} + \sum_{(u,v)\in \overline{E'}} log\frac{1-p_c}{1-p_{u,v}} \qquad (\mathbf{15})$$

Setting the weight of each edge (u,v) to $\frac{p_c}{p_{u,v}} > 0$ and the weight of each non-edge (u,v) to $\frac{1-p_c}{1-p_{u,v}} < 0$, one concludes that the score of H is simply its weight.

Biclustering based on PAttern Mining Software (BicPAMS)

BicPAMS (Henriques et al., 2017) is an aggregate of state-of-the-art pattern mining approaches to the biclustering problem. *BicPAMS* is the most recent pattern mining algorithms, an improved version of prior pattern mining biclustering algorithms since the initial publication of *BicPAM (Henriques and Madeira, 2014a).* Other prior versions of pattern-mining biclustering algorithms that it extends include *BicSPAM (Henriques and Madeira, 2014b) (reviewed in (Pontes et al., 2015a)), BiP (Henriques and Madeira, 2015),* and *BicNET (Henriques and Madeira, 2016). BicPAMS* is a highly parametrized algorithm including parameters relating to coherence of biclusters, structure of biclusters, quality of biclusters, and efficiency of the program. *BicPAMS* was not reviewed in *(Pontes et al., 2015a).* It is a non-heuristic based algorithm. [19]

UniBic

UniBic is an extension/improvement of the graph-based biclustering method: *QUBIC (Li et al., 2009).* In *QUBIC,* the input data matrix is initially transformed to a discrete integer rank matrix prior to subsequent operations. A graph G is constructed based on this matrix in which nodes represent the rows (genes) and the edge weights are number of corresponding conditions (columns) between two genes (rows). The biclustering problem is translated to finding heavy subgraphs in G. [19]

UniBic (Wang et al., 2016) is very similar to *QUBIC* with the exception of edge weight calculation. UniBic applies the longest common subsequence (LCS) algorithm to translate the input data matrix to a rank matrix in which the rows are discretized as rank vectors. The n^{th} smallest value in each row is replaced with the integer n, with priority in ties given to the leftmost value. Edge weight in the graph is calculated as the magnitude of the maximal *LCS* between nodes. *UniBic* demonstrates a strong resilience to noise and can detect biclusters of both shifting and scaling patterns. [19]

Algorithm(Year)	Algorithm Type	Deterministic/ Metric based	Implementation Source
CC (2000)	Greedy Search	Yes / Yes	Python (Eren,2013)
ISA (2003)	Linear Algebra	Yes / No	R package isa2 (Csardi et al. 2010)
OPSM (2003)	Optimal Reordering	Yes / No	BicAT (Barkow et al., 2006)
FLOC (2005)	Stochastic Greedy Search	No / Yes	R package BicARE (Gestraud, 2008)
Bimax(2006)	Graph-Based	Yes / No	Python (Eren,2013)
FABIA (2010)	Generative Biclustering	No / No	Bioconductor fabia (Hochreiter et al., 2010)
PPM (2015)	Probabilistic	No / No	JAVA (Chekouo and Murua, 2015)
UniBic (2016)	Graph-Based	No / No	C (Wang et al., 2016)
BicPAMS (2017)	Pattern-based	No / No	Bicpams.com (Henriques et al., 2017)

➢ **Table 2. 2 Biclustering Algorithm Summary**

3.6 Validation of bicluster solutions

Bicluster solution set M can be defined as a set of biclusters as follows: [19]

$$M = \{\beta_1, \beta_2, \dots, \beta_k\} \quad \textbf{(16)}$$

Where B_k denotes the k^{th} bicluster, $k=1,\dots,K$. A bicluster is a combination of two subsets; one is a subset of objects and the other is a subset of features. Therefore, a bicluster B_k can be represented as:

$$B_k = (O_k, F_k) = \{(X_i, Y_j) | x_i \in O_k, y_j \in F_k\} \quad \textbf{(17)}$$

Where O_k and F_k are subsets of objects and features, respectively, and x_i and y_j denote the i^{th} row and j^{th} column of X, respectively. Size of a bicluster B_k is defined as: [19]

$$|B_k| = |O_k|F_k| \quad \textbf{(18)}$$

Where O_k and F_k are the number of objects and features corresponding to B_k, respectively.

3.6.1 Internal Indices

Internal indices of bicluster solutions also use information only intrinsic to the dataset and the bicluster solution.

Average Residue

Cheng and Church (2000) define *residue* of an observed value x_y in the bicluster B_k as: [19]

$$r_{ij}^{(k)} = x_{ij} - \frac{1}{|O_k|}\sum_{x_i \in O_k} x_{ij} - \frac{1}{|F_k|}\sum_{y_j \in F_k} x_{ij} + \frac{1}{|B_k|}\sum_{(x_i, y_j) \in B_k} x_{ij} \quad \textbf{(19)}$$

Then, they evaluate each bicluster by the *mean squared residue* which is defined as: [19]

$$MSR(B_k) = \frac{1}{|B_k|}\sum_{(x_i, y_j) \in B_k} r_{ij}^{(k)} \quad \textbf{(20)}$$

Yang et al. (2002) introduce the average residue to evaluate the total bicluster solution.

$$ASR = \frac{1}{K}\sum_{k=1}^{K} MSR(B_k) \quad (\mathbf{21})$$

Also, *Madeira and Oliveira (2004)* introduce the residue of overlapping biclusters with the general additive model or the general multiplicative model to evaluate the bicluster solutions. As the average residue becomes close to 0, the bicluster solution is highly evaluated.

$\bar{\Gamma}$ Index

Santamaría et al. (2007) propose an index by imitating the normalized Hubert's statistic *(Jain and Dubes, 1988)*. Let $\boldsymbol{P = (P_{ij})}$ be the proximity matrix of objects so that $\boldsymbol{p_{ij}}$ denotes the distance between two objects $\boldsymbol{x_i}$ and. $\boldsymbol{x_j}$ Also, let $\boldsymbol{C= (c_{ij})}$ be the membership matrix that: [19]

$$C_{ij} = \frac{1}{1 + k_{ij}} \quad (\mathbf{22})$$

Where $\boldsymbol{k_{ij}}$ is the number of biclusters which two objects $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ simultaneously belong to. Then, they define the statistic of objects as: [24]

$$\bar{\Gamma}_O = \frac{2}{n(n-1)} \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(p_{ij} - \mu_p)(c_{ij} - \mu_c)}{\sigma_p \sigma_c} \quad (\mathbf{23})$$

Where $\boldsymbol{\mu_p(\mu_c)}$ and $\boldsymbol{\sigma_p(\sigma_c)}$ are the mean and the standard deviation of $\boldsymbol{P(C)}$, respectively. In the same way, the statistic of features $\boldsymbol{\Gamma_F}$ can be formulated. Then, they define the $\boldsymbol{\Gamma}$ index by combining the two statistics as follows:

$$\bar{\Gamma} = \frac{n\bar{\Gamma}_O + m\bar{\Gamma}_F}{n + m} \quad (\mathbf{24})$$

Since the numerator increases as similar objects or features are grouped together, a bicluster solution with large $\boldsymbol{\Gamma}$ is preferred in the range of [-1, 1].

3.6.2 External Indices

External indices of biclustering are used to compare two bicluster solutions. If we have prior grouping information, we can evaluate a bicluster solution by comparing with the known information. Let $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ be bicluster solutions which consist of $\boldsymbol{K_1}$ and $\boldsymbol{K_2}$ biclusters, respectively. We consider that one of them is the obtained solution and the other is the prior solution. Then, we can denote each bicluster solution as : [19]

$$M_j = \left\{B_1^{(j)}, B_2^{(j)}, \dots, B_{K_j}^{(j)}\right\}, \quad j = 1, 2 \quad \mathbf{(25)}$$

$$\text{where} \quad B_k^{(j)} = (O_k^{(j)}, F_k^{(j)}).$$

Prelić et al., (2006) propose the external index based on the *Jaccard index (Downton and Brennan, 1980).* The *Prelić* index compare two solutions based on categorization of objects as follows: [19]

$$I_{Prelic}(M_1, M_2) = \frac{1}{K_1}\sum_{i=1}^{K_1} \max_j J\left(O_i^{(1)}, O_j^{(2)}\right) \quad (\mathbf{26})$$

Where J (A,B) is the *Jaccard index* for two sets A and B: [19]

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (\mathbf{27})$$

Liu and Wang (2007) propose another external index which compares two solutions considering both objects and features. Their index (the LW index) can be formulated as: [19]

$$I_{LW}(M_1, M_2) = \frac{1}{K_1}\sum_{i=1}^{K_1} \max_j \frac{|O_i^{(1)} \cap O_j^{(2)}| + |F_i^{(1)} \cap F_j^{(2)}|}{|O_i^{(1)} \cup O_j^{(2)}| + |F_i^{(1)} \cup F_j^{(2)}|} \quad (\mathbf{28})$$

Whereas above two indices are based on the *Jaccard index, Santamaría et al. (2007)* propose an external index based on the *Dice index (Dice, 1945)* which is called the F_1 measure by *Turner et al.(2005)* in biclustering cases. *The Santamaría index* computes the overall relevance of two bicluster solutions as follows: [18]

$$I_{Santamaria}(M_1, M_2) = \frac{1}{K_1}\sum_{i=1}^{K_1} \max_j D\left(B_i^{(1)}, B_j^{(2)}\right) \quad (\mathbf{29})$$

Where ***D (A,B)*** is *the Dice index* given by: [19]

$$D(A,B) = \frac{2 \times |A \cap B|}{|A| + |B|} \quad (\mathbf{30})$$

Above three indices lie in the range of [0,1]. The indices which are close to 1 mean that the two bicluster solutions are similar to each other. While *the Prelić index* compares only object sets and the LW index compares object sets and feature sets independently, the Santamaría index compares two solutions using pairs of objects and features. Therefore, *the Santamaría index* is the most conservative index among above three indices

4 Conclusion

The biclustering of microarray data has been the subject of a large research, no one of the existing biclustering algorithms is perfect then others, it can be fast, or accurate or easy in implementation, give more result (biclusters), delete the noise data, algorithms are different.

we know also that biclustering algorithms are NP-hard problems and complicated search problems so metaheuristic methods are very good choices for solving.

the construction of biclusters for large microarray data is still a problem that requires a continuous work, even in the case of accurate and rapid results, we will face problem in interpretation of the output in the case of a large amount of information

CHAPTER 3
PROPOSED APPROACH

CHAPTER 3 PROPOSED APPROACH

1 Introduction

We will present in this chapter new model based on Cheng and Church and Kmeans algorithms. They uses a simple data model reflecting the fundamental idea of clustering and biclustering, while aiming to determine all optimal biclusters in reasonable time. This model has the benefit of providing a basis to investigate the usefulness of the biclustering concept in general, independently of interfering effects caused by Cheng and Church and Kmeans algorithms hybridization, and the effectiveness of more complex scoring schemes and biclustering methods. We will call this proposed approach as *KMCC*.

KMCC model for biclustering based on re-ordering of data matrices uses clustering algorithm in one dimension to define submatrices that are then optimally re-ordered in the other dimension to generate biclusters. Several different objective functions can be used to quantify the degree of similarity between adjacent rows and columns in the final arrangement

2 Cheng and Church Algorithm

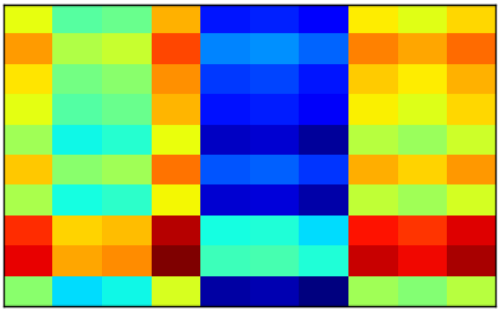
Cheng and Church "CC" were the first to introduce biclustering to gene expression analysis, searches for bicluster with constant values, rows or columns. The algorithm divides an expression model into three parts: attribute residue, object residue, and δ-cluster residue (or background residue). Their algorithmic framework represents the biclustering problem as an optimization problem, defining a score for each candidate bicluster and developing heuristics to solve the constrained optimization problem defined by fined by this. [20]

Starting from an adjusted matrix, the mean squared residue of the bicluster is:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i\in I, j\in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \qquad \textbf{(31)}$$

The residue is meant to measure how an element differs from the row mean, column mean, and overall mean of the bicluster. If all the elements of the bicluster have small residues, clearly the mean squared residue will be small. [20]

H(I,J) achieves a minimum when all the rows and columns of the bicluster are shifted versions of each other. In other words, if we can represent every element of the bicluster as $\boldsymbol{a_{ij}=r_i+c_j}$, where ***r*** is a column vector with $|\boldsymbol{I}|$ entries and ***c*** is a row vector with $|\boldsymbol{J}|$ entries, then the score of ***M*** will be ***0***, to visualize these shifted rows and columns, here is a matrix with a perfect mean squared residue [18]:



➢ **Figure 3. 1 CC Biclusters data visualization**

The lowest score ***H(I, J) = 0*** indicates that the gene expression levels fluctuate in unison. This includes the trivial or constant biclusters where there is no fluctuation. These trivial biclusters may not be very interesting but need to be discovered and masked so that more interesting ones can be found. The row variance may be an accompanying score to reject trivial biclusters: [20]

$$V\ (I,J)\ =\ \frac{1}{|J|}\sum_{j\in J}(eij\ -\ eIj\ )^2 \tag{32}$$

The higher the value of ***H*** is, the more disordered the data is. In Cheng and Church algorithm, a greedy method is used to select submatrix with a low ***H*** score. It is divided into two phases. Firstly, the method remove the row or column to achieve the largest decrease of the score. For the current submatrix, they calculate the average residue score of each row using $d(i) = \frac{1}{|J|}\sum_{j\in J} RS_{IJ}(i,j)$ and the average residue score of each column using $e(j) = \frac{1}{|I|}\sum_{i\in I} RS_{IJ}(i,j)$, then choose the row or column with the maximal score and delete it from the current submatrix, until ***H(I, J) < δ***. Also they use a parameter ***α***, so that they can delete a set of nodes each time before the score is recalculated. Without updating the score after the removal of each node, the matrix may shrink too much and one may miss some large ***δ***-clusters. One may also choose an adaptive α based on the score and the size during the iteration. Secondly, they add rows and columns so that the matrix with the maximal size can be obtained. [20]

Cheng and Church tries to find biclusters that are as large as possible, with the restriction that their ***H*** score must be less than some threshold ***δ***. Like most biclustering problems, this one is NP-hard. Therefore, the method proceeds via a simple greedy approach: it starts with the largest possible bicluster, then removes rows and columns that most reduce ***H(I,J)***.

As Cheng and Church prove in their paper, this greedy removal can be done efficiently, without the need to recalculate ***H*** for every possible row and column removal. To do so, we define the mean squared residue of any row ***i*** or any column ***j*** in the bicluster. [20]

Then this part of the algorithm, the single node deletion step, proceeds as follows:

Input: data matrix **A** = *(X, Y)*,maximum acceptable MSR δ ≥ 0

Output: δ-bicluster (*I, J*) with *H*(*I, J*) ≤ *δ*

while *H*(*I, J*) > δ **do**

Find the row $i \in I$ with the largest $d(i) = \frac{1}{|J|}\sum_{j\in J} r(a_{ij})^2$

Find the column $j \in J$ with the largest $d(j) = \frac{1}{|I|}\sum_{i\in I} r(a_{ij})^2$

Update (*I,J*) by removing the row or column with the largest *d()*

➢ **Figure 3. 2 Single Node Deletion**

In order to speed up node deletion, Cheng and Church also uses a method to remove multiple rows and columns at once. This multiple node deletion step proceeds as follows:

Input: data matrix **A** = *(X, Y)*,maximum acceptable MSR δ ≥ 0, threshold for multiple node deletion α > 1

Output: δ-bicluster (*I, J*) with *H*(*I, J*) ≤ *δ*

(*I, J*) = (*X, Y*)

if *H*(*I, J*) ≤ δ **then**

return (*I, J*)

repeat

Remove the rows $i \in I$ with $\frac{1}{|J|}\sum_{j\in J} r(a_{ij})^2 > \alpha H(I,J)$

Remove the columns $j \in J$ with $\frac{1}{|I|}\sum_{i\in I} r(a_{ij})^2 > \alpha H(I,J)$

until there is no update in (*I, J*)

Switch to algorithm Single Node Deletion

- **Figure 3. 3 Multiple Node Deletion**

Multiple and single node deletion stop when $\boldsymbol{H(I,J) <= \delta}$. At this point, the algorithm tries a node addition step to add any rows or columns that do not make $\boldsymbol{H(I,J)}$ worse.

The node addition algorithm proceeds as follows:

Input: data matrix **A** = *(X, Y),* δ-bicluster (*I, J*)

Output: bicluster (*I', J'*) with $H(I', J') \leq H(I, J)$, $I' \supseteq I$, and $J' \supseteq J$

repeat

add the columns $j \notin J$ with $\frac{1}{|I|}\sum_{i\in I} r(a_{ij})^2 \leq H(I,J)$

add the rows $i \notin I$ with $\frac{1}{|J|}\sum_{j\in J} r(a_{ij})^2 \leq H(I,J)$

// Add inverted rows into the bicluster that form "mirror images"

add the rows $i \notin I$ with $\frac{1}{|J|}\sum_{j\in J}(-a_{ij} + a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I,J)$

until there is no update in (*I,J*)

- **Figure 3. 4 Single node addition**

After node addition, the bicluster is added to the list of results and the algorithm starts again from the beginning. However, because this is a deterministic procedure, it would just find the same bicluster again. Therefore, after finding a bicluster, its entries in the original data are replaced by entries drawn from a uniform random distribution over the range determined by the minimum and maximum of the original dataset. The whole algorithm looks like this:

Cheng-Church*(***A** , *δ):*

Input: data matrix **A** = *(X, Y),* **δ:** *maximal mean square residue score*.

Output: bicluster (*I, J*)

Define $e_{Ij} = \frac{1}{|I|}\sum_{i\in I} e_{ij}$

Define $e_{iJ} = \frac{1}{|J|}\sum_{j\in J} e_{ij}$

Define $e_{IJ} = \frac{1}{|I||J|}\sum_{i\in I, i\in J} e_{ij}$

Define $RS_{IJ}(i,j) = e_{ij} - e_{iJ} - e_{Ij} + e_{IJ}$

Define $H(I,J) = \sum_{i\in I, j\in J} \frac{RS_{ij}^2}{|I||J|}$

Initialize a bicluster (*i, j*) *with I=U, J=V.*

Deletion phase:

While *(H(I,J) > δ)* do

Compute for $i \in I$, $d(i) = \frac{1}{|J|} \sum_{j \in J} RS_{I,J}(i,j)$

Compute for $j \in J$, $e(j) = \frac{1}{|I|} \sum_{i \in I} RS_{I,J}(i,j)$

If $max_{i \in I}\, d(i) < max_{j \in J}\, e(j)$ assign $I = I \setminus \{argmax_i(d(i))\}$.

Else $J = J \setminus \{argmax_j(e(j))\}$

Addition phase:

assign $I' = I$, $J' = J$

While $(H(I',J') > \delta)$ do

Compute for $i \in U \setminus I$, $d(i) = \frac{1}{|J|} \sum_{j \in J} RS_{I,J}(i,j)$

Compute for $j \in V \setminus J$, $e(j) = \frac{1}{|I|} \sum_{i \in I} RS_{I,J}(i,j)$

If $max_{i \in I}\, d(i) < max_{j \in J}\, e(j)$ assign $I' = I \cup \{argmax_i(d(i))\}$.

Else $J' = J \cup \{argmax_j(e(j))\}$

Return *I,J*

- **Figure 3. 5 The Cheng-Church algorithm for finding a single bicluster**

To discover more than one bicluster, Cheng and Church suggested repeated application of the biclustering algorithm on modified matrices. The modification includes randomization of the values in the cells of the previously discovered biclusters, preventing the correlative signal in them to be beneficial for any other bicluster in the matrix. The whole algorithm: [20]

Cheng-Church*(A , δ, n)*:

Input: data matrix **A** = *(X, Y)*, **δ:** *maximal mean square residue score*, ***n**: number of biclusters*

Output: List of Biclusters *L*

Preprocess the missing values of A

List *L*

Bicluster *B*

Repeat n times

$B = A$ *//initialize B to all rows and all columns*

B_δ = multiple node deletion phase(B, δ)

B'_δ = single node deletion phase(B_δ, δ)

B''_δ = single node addition phase(B'_δ, δ)

$L = L \oplus B''_\delta$ // append result *B* to *L*

Substitution phase (B''_δ,A)

End repeat

Return *L*

- **Figure 3. 6 The Cheng-Church algorithm**

2.1 Parameter selection

In Cheng and Church algorithm, there are two important parameters δ and α that need to be set before the algorithm running, where δ is a threshold of score function H and measures the extent of data consistency. The parameter δ influences the quality of matrix clustering and in general it is better if the value is smaller. However, if δ is too small, the scale of the submatrix will be over small and easy to lose information. Hence, a balance point should be found for this parameter before running the algorithm. The parameter α is used in the deletion course of the first phase in the original algorithm, which is also an important threshold. It directly influences the clustering speed. [20]

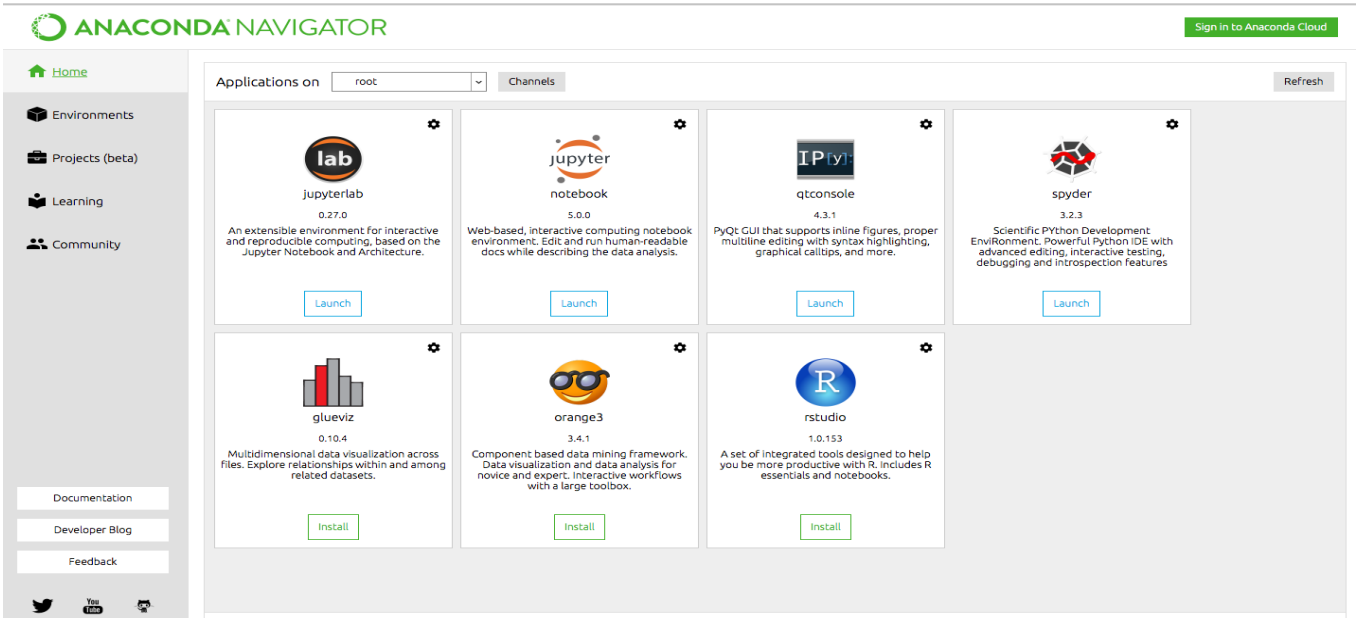
3 Tools

Explain the software, tools and programming languages we have used to get the results:

3.1 Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. [21]

Anaconda distribution comes with more than 1,500 packages as well as the Conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI). [21]

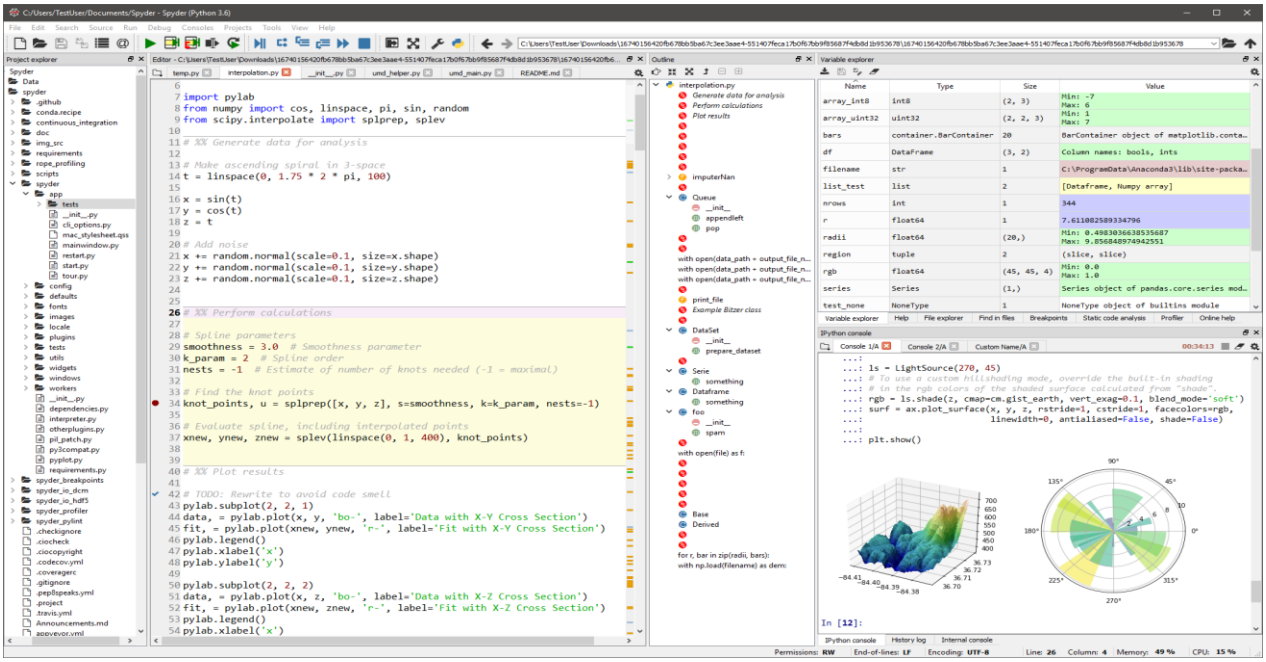


➢ **Figure 3. 7 Anadconda interface**

3.2 Spyder

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package. [22]

Beyond its many built-in features, its abilities can be extended even further via its plugin system and API. Furthermore, Spyder can also be used as a PyQt5 extension library, allowing developers to build upon its functionality and embed its components, such as the interactive console, in their own PyQt software. [22]



➢ **Figure 3. 8 Spyder interface**

3.3 Python

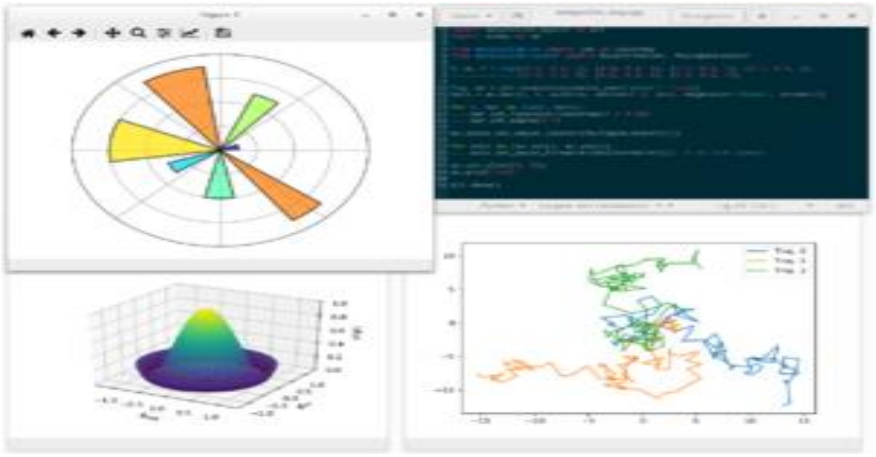
Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. [23]

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. [23]

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council. [23]

3.3.1 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. [24]



➢ **Figure 3. 9 Matplotlib**

3.3.2 NumPy

NumPy is the fundamental package for scientific computing with Python

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. [25]

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

3.3.3 Pandas

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

pandas is a NumFOCUS sponsored project. This will help ensure the success of development of pandas as a world-class open-source project, and makes it possible to donate to the project. [26]

3.4 Datasets

Gene expression data is usually arranged in a matrix such that each row represents a gene and each column corresponds to an experimental condition.

Dataset Name	Selected files	Used name	Shape	Biclusters
Synthetic dataset1[1] [27]	Art_Noise0d1_Set1.txt	ArtNd1	(200x50)	3
	Art_Noise0d5_Set1.txt	ArtNd5		
Synthetic dataset2 [19]	narrow_100_10_data1.txt	Narf1	(1000x100)	4
	narrow_100_10_data2.txt	Narf2		
Synthetic dataset3 [28]	OrderPreservingS500NormalN0M0I0.txt	Ordf1	(500, 50)	3
	OrderPreservingS500NormalN0M0I1.txt	Ordf2		
Synthetic dataset4 [19]	square_15_15_typeII_data1.txt	Sqrf1	(150, 100)	4
	overlap_3_3_data1.txt	Ovrf1	(200, 150)	

➢ **Table 3. 1 Used datasets**

[1] That microarray dataset have noise density from low density d=1 to high density d=5.

4 Evaluation measures

Before discussing the improvements we made, we first outline the score evaluation measure of this model.

4.1 Clustering Error(CE)

Consider subspace clustering's $S = \{S_1, S_2, \ldots, S_K\}$ and $S' = \{S'_1, S'_2, \ldots, S'_K\}$ of K and K' clusters, respectively. Since the subspace clustering's are not partitions of the data matrix elements, we cannot form a confusion matrix M. Instead, let us define *the cluster intersection matrix* $T=(t_{ij})$ as a $K \times K'$ matrix in which tij is the number of data matrix elements shared by the clusters S_i and S' j. More formally, $t_{ij} = |supp(S_i) \cap supp(S'_j)|$. [29]

Let us transform T into a square matrix by adding rows or columns of zeroes if necessary and use the Hungarian method to find a permutation of the cluster labels such that the sum of the diagonal elements of T is maximized. [29]

Denote this maximized sum by D_{max}. Now, we define the clustering error (CE) for subspace clustering's as: [29]

$$CE(S, S') = \frac{|U| - D_{max}}{|U|}$$

4.2 Relative Non-Intersecting Area (RNIA)

If we want to compare a subspace clustering S to a true clustering S', a simple approach would be to calculate the *precision*, the *recall*, and the *F-measure*, used widely in the information retrieval literature to measure the success of the retrieval task. Retrieval is similar to subspace clustering in that it aims to extract a subset of the data that is alike in some respect, while the rest of the data is not assumed to be grouped in any way. Hence, a subspace clustering is like the unsupervised retrieval of several disjoint groups. [29]

Using our subspace clustering notation, recall is defined as $|I|/supp(S')$; it measures how big part of the matrix elements of the true clustering S' is retrieved (covered) by the clustering S. Precision is defined as $|I|/supp(S)$; it measures the proportion of the matrix elements in the clustering S that belong to the true clustering S'. The F-measure is just the geometric mean of the precision and the recall. [29]

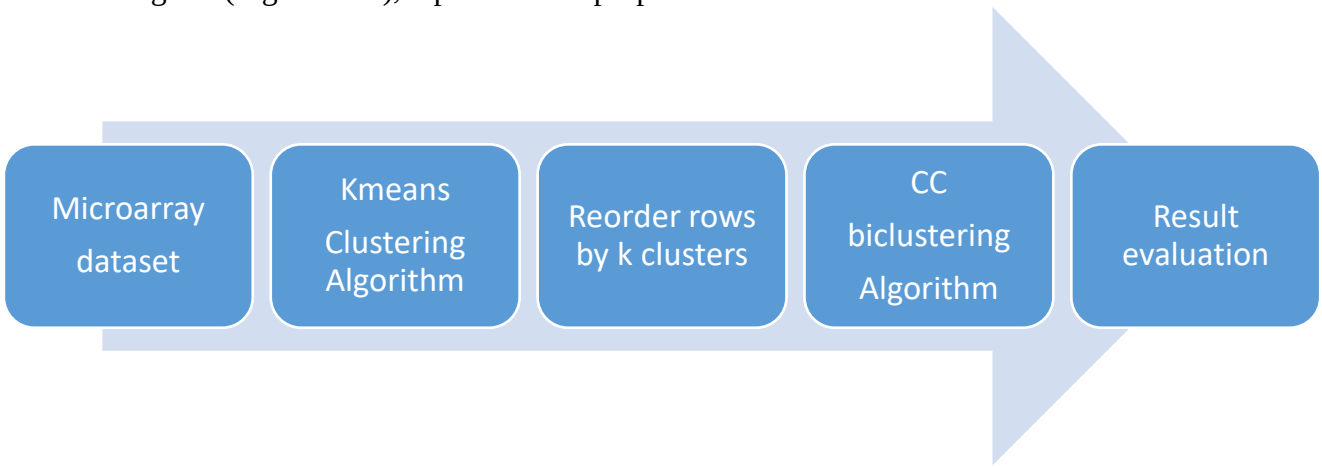
A big drawback of these measures is that they are not symmetric. A symmetric alternative is the relative non-intersecting area $(RNIA)^3$ of the two clustering's:

$$RNIA(S, S') = \frac{|U| - |I|}{|U|}$$

We will focus on CE and RNIA in the rest of this dissertation, since these two measures have more desirable properties than other measures. [29]

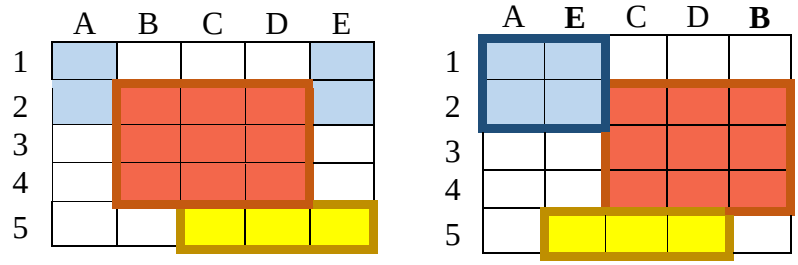
5 The KMCC model

Next diagram (Figure 3.10), represents the proposed model workflow:



➢ **Figure 3. 10 *KMCC* Model diagram**

We will use the *Kmeans* algorithm to reordering the matrices (see figure 3.11), then apply the *CC* algorithm and compare the score of external evaluation measures, the runtime of algorithm before and after reordering.



(a) (b)

➢ **Figure 3. 11 Biclustering example with three clusters illustrating the reordering problem.**

Figure 3.11 (a) shows the example matrix where the red and the yellow bicluster form contiguous blocks (thick borders), but the blue bicluster is split into two unconnected blocks. Figure 3.11 (b) by reordering the columns, the blue bicluster becomes contiguous.

The proposed model Algorithm:

KMCC(*A, δ, k, n*)

Input: data matrix ***A*** *= (X, Y),* **δ:** *maximal mean square residue score, n: number of biclusters*

Output: List of Biclusters *L*

List *L*

List *c* // list of clusters

Bicluster *B*

data matrix *AK*

Kmeans(*A, k*):

Place the centroids *c_1, c_2, ... c_k* randomly

Repeat until convergence

for each data point *x_i*:

- find the nearest centroid(*c_1, c_2, ... ,c_k*)
- assign the point to that cluster

for each cluster *j = 1..k*

- new_centroid = mean of all points assigned to that cluster

End repeat

Return *c*

for each cluster *c*

for each row of the matrix *A*

If the cluster of current row == cluster *c*

- assign current row to the new matrix *AK*

Repeat n times

B = AK *//initialize B to all rows and all columns*

B_δ = multiple node deletion phase(*B*, δ)

B'_δ = single node deletion phase(B_δ, δ)

B''_δ = single node addition phase(B'_δ, δ)

$L = L \oplus B''_\delta$ // append result *B* to *L*

Substitution phase (B''_δ,*AK*)

End repeat

Return *L*

➢ **Figure 3. 12 The proposed model Algorithm**

6 Result discussion

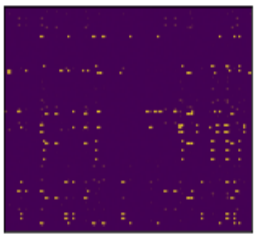
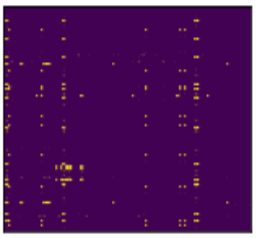
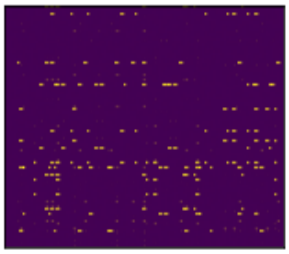
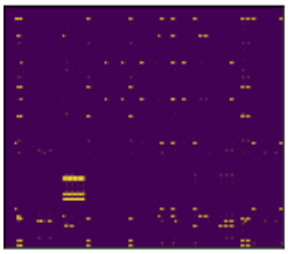
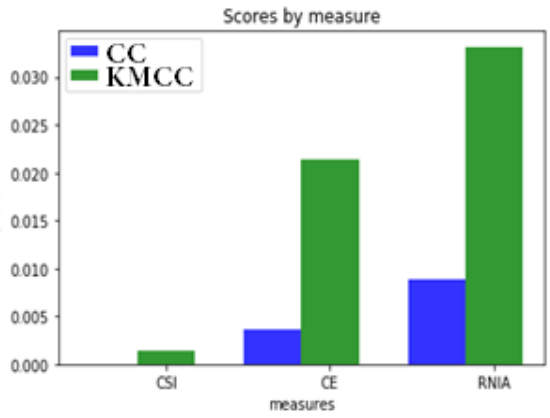
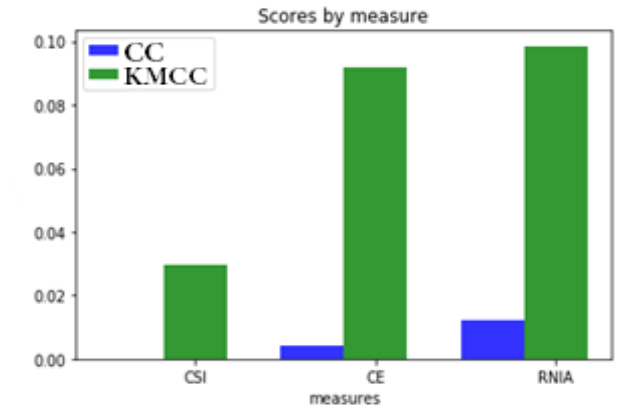
In this section, we present the results for our proposed model for a variety of interesting datasets. We first demonstrate the effectiveness of the proposed model by analyzing datasets. For this purpose, we chose to examine a small synthetic gene expression data matrix, then apply the proposed model to larger synthetic gene expression data with different biclusters forms Narrow, square and overlap, For each of these data sets, we draw comparisons of results.

We have introduced KMCC Model above, that Kmeans seeks clusters such that all rows have some minimum correlation with all other rows, in this step the result matrix of microarray will reordered with selected number of clusters k. We will compare results with k of Kmeans and δ of CC, and we will save other parameters as default. We determined the value ranges through experiments to provide referable information for realizing adaptive setting for the parameters.

Case Study 1:

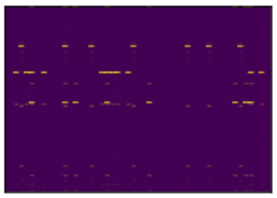
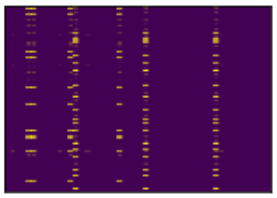
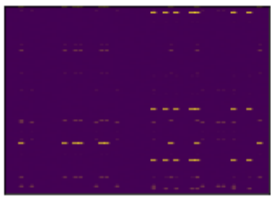
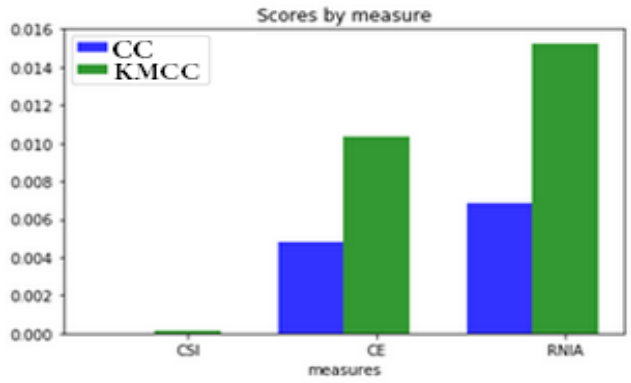
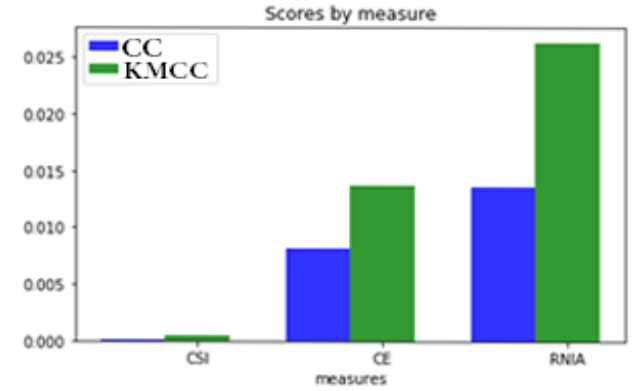
The proposed method was tested on synthetic dataset with different noise degree, we are selected 11 datasets, genes (the rows of the data matrix) conditions (columns of the data matrix), In *Kmeans algorithm*, there are an a important parameters *k* that need to be set before the algorithm running, where *k* is a number of clusters. In *CC algorithm*, there are two important parameters δ and α that need to be set before the algorithm running, where δ is a threshold of score function *H* and measures the extent of data consistency.

In this proposed model, using α =1.2 and δ =1200 with same parameters as original paper [20], we will run *Kmeans* with *k* = 6 in first case study ,with *k* = 100 in second study case, the green bars mark the *KMCC* model score in parallel with the bleu bar mark the original CC algorithm score of three external evaluation measures is Relative Non-Intersecting Area (RNIA) and Clustering Error (CE), Campello Soft Index (CSI).

CC	KMCC	CC	KMCC
			
			

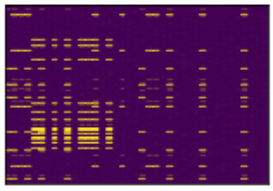
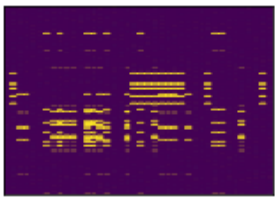
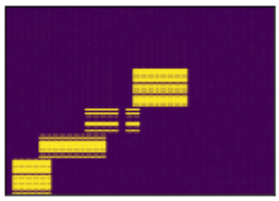
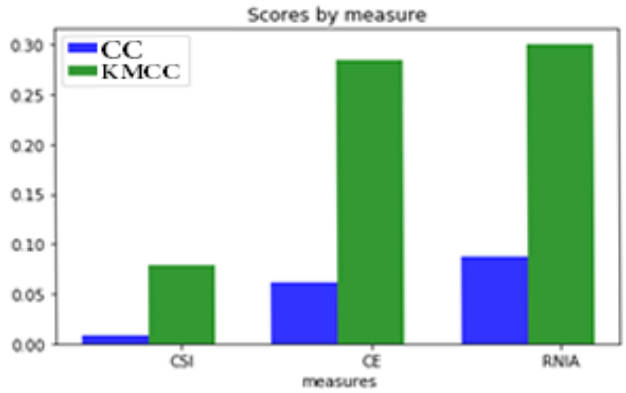
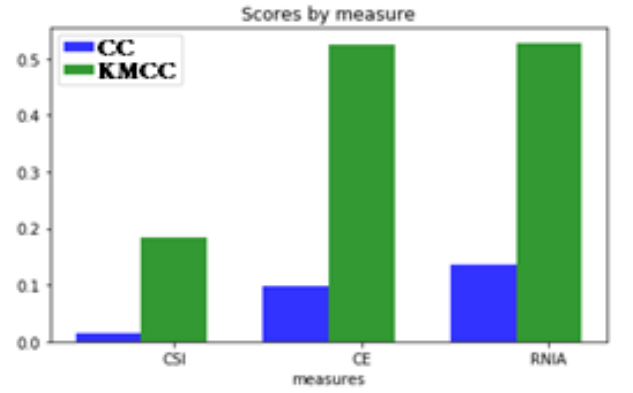
Narf1				Narf2			
	CC	KMCC	increase		CC	KMCC	increase
CSI	0.000	0.0001	0.0001	CSI	0.000	0.030	0.030
CE	0.004	0.021	0.017	CE	0.005	0.006	0.001
RNIA	0.09	0.033	0.024	RNIA	0.008	0.010	0.002
Runtime	10.70 s	5.98 s		Runtime	7.74 s	5.27 s	

➢ **Table 3. 2 CC vs KMCC with Synthetic dataset1, k=6**

CC	KMCC	CC	KMCC
			
			

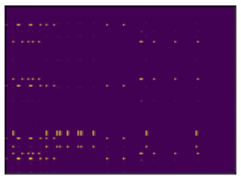
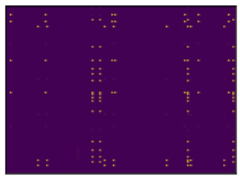
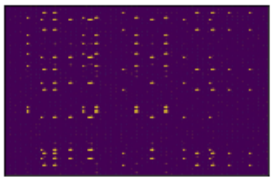
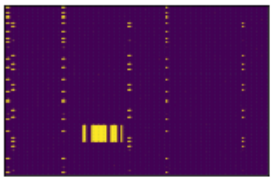
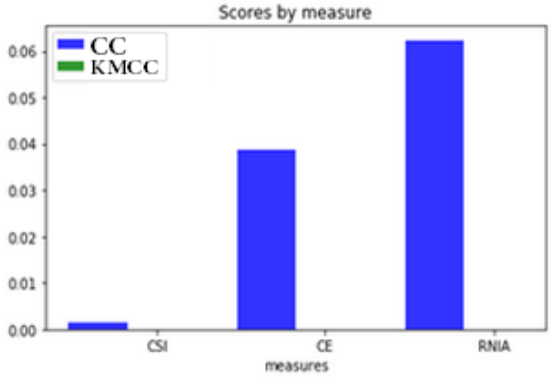
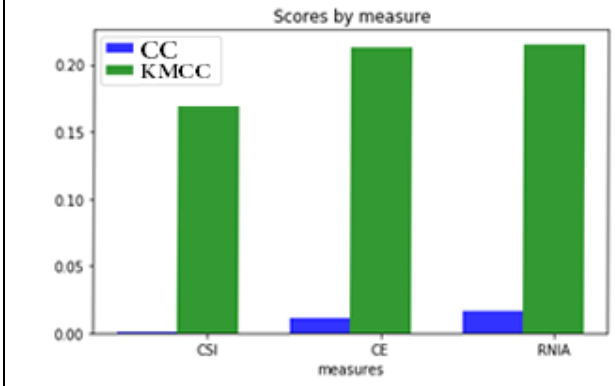
Ordf1				Ordf2			
	CC	KMCC	increase		CC	KMCC	increase
CSI	0.000	0.000	0.000	CSI	0.000	0.000	0.000
CE	0.005	0.010	0.005	CE	0.008	0.014	0.006
RNIA	0.007	0.015	0.008	RNIA	0.014	0.026	0.012
Runtime	0.38 s	0.28 s		Runtime	0.37 s	0.28 s	

➢ **Table 3. 3 CC vs KMCC with Synthetic dataset2, k=6**

CC	KMCC	CC	KMCC
			
			

ArtNd1				ArtNd2			
	CC	KMCC	increase		CC	KMCC	increase
CSI	0.004	0.151	0.147	CSI	0.013	0.184	0.171
CE	0.041	0.486	0.445	CE	0.099	0.526	0.427
RNIA	0.073	0.488	0.415	RNIA	0.137	0.528	0.319
Runtime	0.38 s	0.28 s		Runtime	0.15 s	0.12 s	

➢ **Table 3. 4 CC vs KMCC with Synthetic dataset3, k=6**

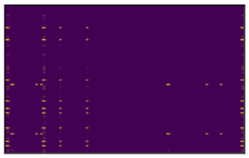
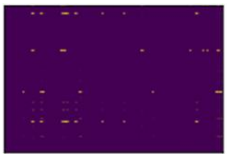
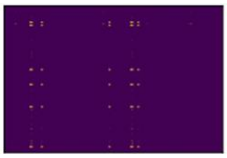
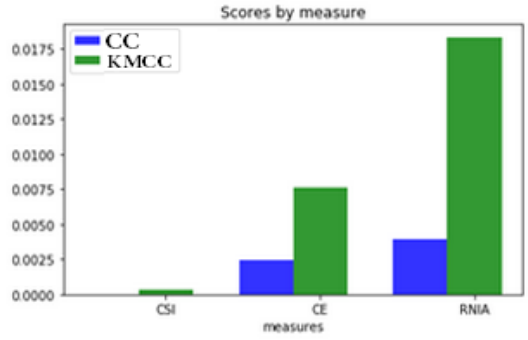
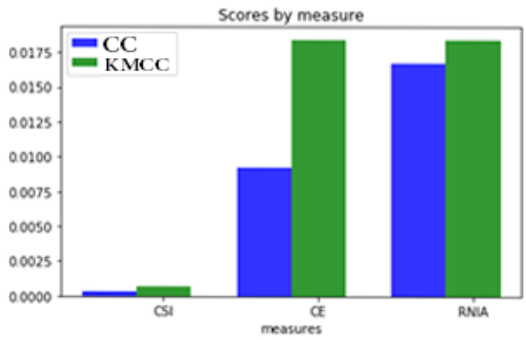
CC	KMCC	CC	KMCC
			
			

Ovrlp1				Sqrf1			
	CC	KMCC	increase		CC	KMCC	increase
CSI	0.001	0.000	0.000	CSI	0.000	0.245	0.245
CE	0.39	0.000	0.000	CE	0.005	0.380	0.375
RNIA	0.062	0.000	0.000	RNIA	0.007	0.380	0.373
Runtime	0.33 s	0.23 s		Runtime	0.19 s	0.09 s	

➢ **Table 3. 5 CC vs KMCC with Synthetic dataset4, k=6**

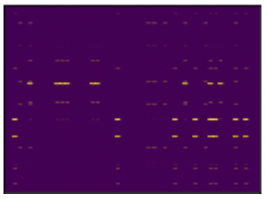
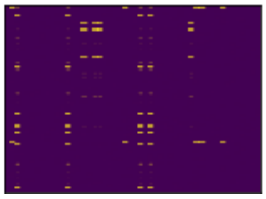
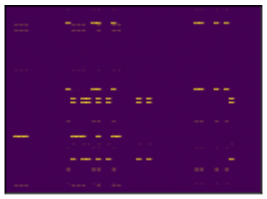
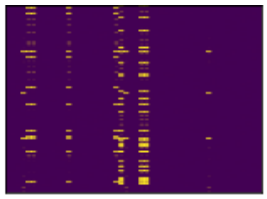
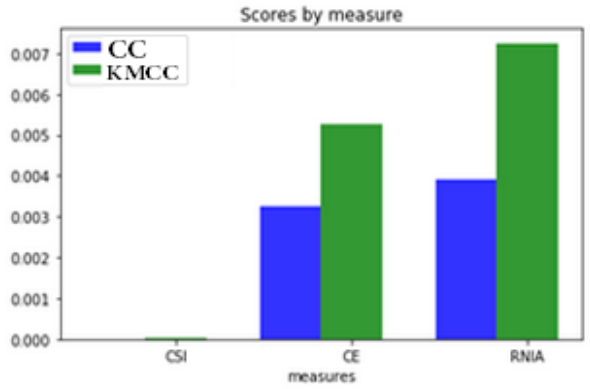
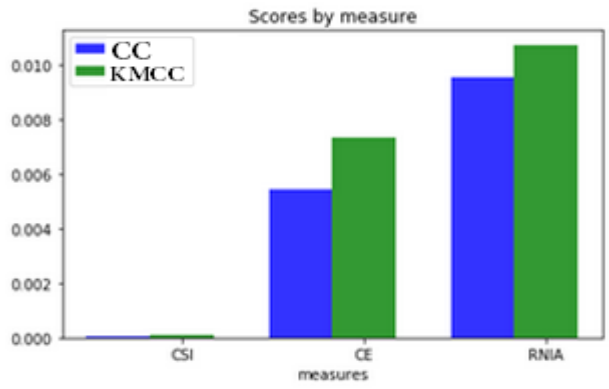
Case Study 2:

As in first case study we fixed value for parameters a =1.2, δ=1200, this time we will try to run *Kmeans* with k = 100.

CC	KMCC	CC	KMCC
			
			

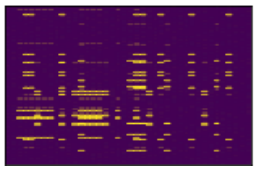
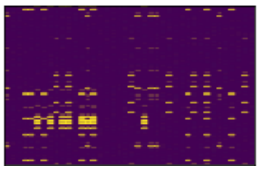
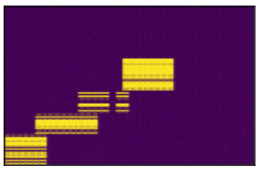
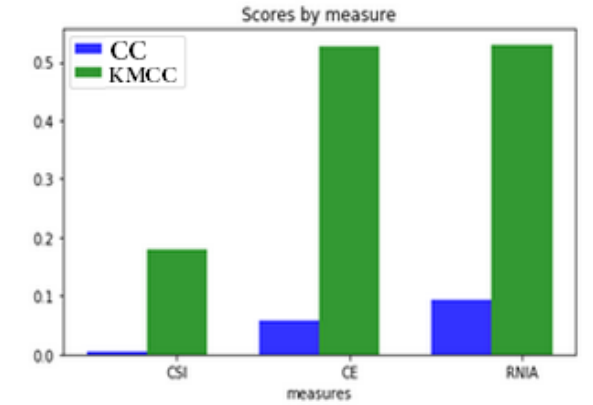
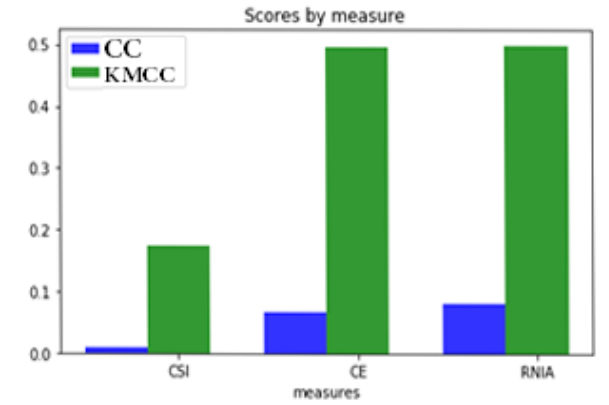
Narf1				Narf2			
	CC	KMCC	increase		CC	KMCC	increase
CSI	0.000	0.000	0.000	CSI	0.000	0.001	0.001
CE	0.002	0.008	0.006	CE	0.009	0.018	0.009
RNIA	0.004	0.018	0.016	RNIA	0.017	0.018	0.001
Runtime	2.41 s	1.92 s		Runtime	7.74 s	5.27 s	

➢ **Table 3. 6 CC vs KMCC with Synthetic dataset1, k=100**

CC	KMCC	CC	KMCC
			
			

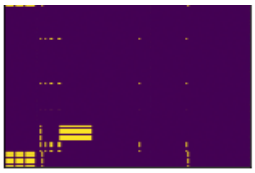
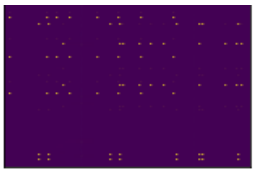
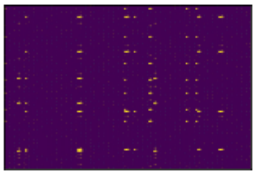
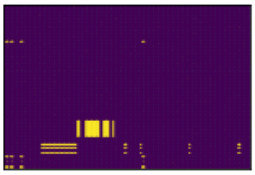
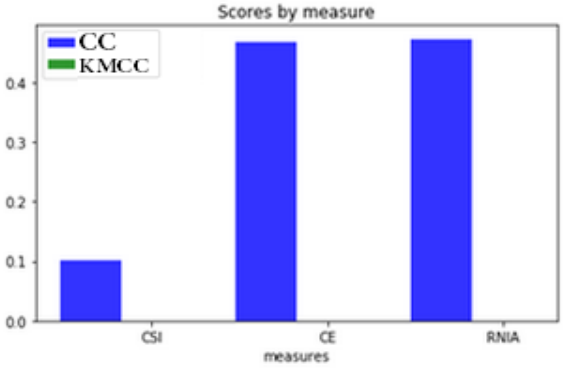
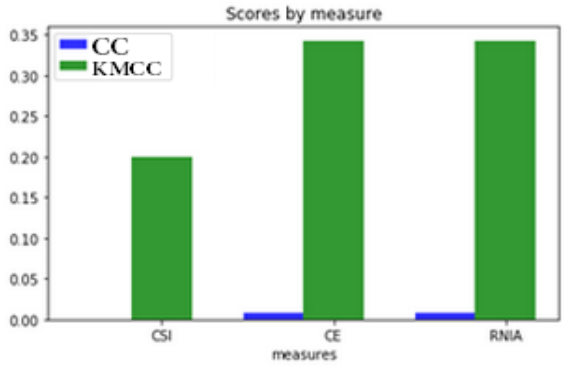
Ordf1				Ordf2			
	CC	KMCC	increase		CC	KMCC	increase
CSI	0.000	0.000	0.000	CSI	0.000	0.000	0.000
CE	0.003	0.005	0.002	CE	0.005	0.007	0.002
RNIA	0.004	0.007	0.003	RNIA	0.010	0.011	0.001
Runtime	0.62 s	0.49 s		Runtime	0.54 s	0.45 s	

➢ **Table 3. 7 CC vs KMCC with Synthetic dataset2, k=100**

CC	KMCC	CC	KMCC
			
			

ArtNd1				ArtNd2			
	CC	KMCC	increase		CC	KMCC	increase
CSI	0.004	0.179	0.175	CSI	0.008	0.175	0.167
CE	0.057	0.527	0.470	CE	0.067	0.498	0.431
RNIA	0.094	0.530	0.436	RNIA	0.081	0.500	0.419
Runtime	0.15 s	0.17 s		Runtime	0.17 s	0.21 s	

➢ **Table 3. 8 CC vs KMCC with Synthetic dataset3, k=100**

CC	KMCC	CC	KMCC
			
			

Ovrlp1				Sqrf1			
	CC	KMCC	increase		CC	KMCC	increase
CSI	0.101	000	0.000	CSI	0.000	0.200	0.200
CE	0.468	000	0.000	CE	0.007	0.343	0.336
RNIA	0.473	000	0.000	RNIA	0.008	0.343	0.335
Runtime	0.35 s	0.23 s		Runtime	0.20 s	0.08 s	

➢ **Table 3. 9 CC vs KMCC with Synthetic dataset4, k=100**

In Case Study one Table 3.2, Table 3.3, Table 3.5 shows the increase of results with three embedded biclusters, and four embedded biclusters in Table 3.4, we ran the algorithm for 20 iterations with every dataset. The improvement in results score is very clear and high in every form of dataset, only in Table 3.5 with overlap biclusters we did not get good results.

In Case Study two with ran with different parameters of *Kmeans* Table 3.6, Table 3.7, Table 3.9 shows the increase of results with three embedded biclusters, and four embedded biclusters in Table 3.8, we ran the algorithm for 20 iterations with every dataset. The improvement in results score is very clear and high in every form of dataset, only in Table 3.9 with overlap biclusters we did not get good results.

7 Conclusion

The application of Cheng and Church biclustering to some a datasets give poor result. Therefore, we proposed new model to improve biclusters results accuracy and runtime. *CC algorithm* is a greedy method essentially. The proposed model *KMCC* reordering of data by *Kmeans* algorithm lead to new results, improves the original *CC algorithm* accuracy score and run time of algorithm, gives good results with various matrix formats. In addition, we can see how the algorithm's running speed has changed and improved.

GENERAL CONCLUSION

Biclustering data is a complex task involving the choice between many different methods, parameters and performance metrics, with implications in many real-world problems. We will present in this dissertation a new model of biclustering algorithm and objectively evaluated their performance, which can often get better biclustering results or faster biclustering speed for some datasets than some classical biclustering algorithms.

KMCC model is robust to noisy data and allows finding biclusters with different shapes. The proposed model is using the *Kmeans* to reordering the microarray rows, then applying the *CC algorithm*. We got great results score in the accuracy of the biclusters, across the synthetic datasets used in our experiment, we determined that *KMCC model* was the best performing algorithm in terms of Relative Non-Intersecting Area (RNIA) and Clustering Error (CE), Campello Soft Index (CSI).

These important results cannot be neglected; in the field of science the accuracy of the results are the main objective of the algorithm, because each cell of the microarray express a real gene. *KMCC model* is sensitive to overlap biclusters.

The next work is to further improve this model algorithm in time cost or accuracy quality and overcome its drawback.

BIBLIOGRAPHY

[1] B. Steven and L. Fran, "Bioinformatics– a new era ,USA," in *Trends guide to bioinformatics.*, Cambridge, Elsevier Science, 1998.

[2] W. C. John et L. S. Herbert, «Catalyzing Inquiry at the Interface of Computing and Biology,» National Academies Press (US), Washington (DC), 2005.

[3] G. Yevgeniy, «Introduction to DNA Microarrays,» [En ligne]. Available: https://bitesizebio.com/7206/introduction-to-dna-microarrays/. [Accès le 10 5 2019].

[4] S. Stoakes, «History of Microarrays,» News Medical, [En ligne]. Available: www.news-medical.net/life-sciences/History-of-Microarrays.aspx,. [Accès le 10 05 2019].

[5] K. Muthukumarasamy and V. Renu, Practical Chemoinformatics, New Delhi: Springer India, 2014.

[6] NCBI Resource Coordinators, «Nucleic Acids Research,» *Database Resources of the National Center for Biotechnology Information* , p. D12–D17, 28 November 2016.

[7] D. Francis, «Types of Observation in the Scientific Method,» 25 April 2017. [En ligne]. Available: www.sciencing.com/types-observation-scientific-method-8295233.html. [Accès le 10 05 2019].

[8] I. Kenichi, I. Yoshikuni et N. Shuji, «Genomic-Wide Analysis with Microarrays in Human Oncology,» *Microarrays*, vol. 4, n° %14, pp. 454-473, 2015.

[9] «How DNA Microarrays Work,» © WGBH Educational Foundation, [En ligne]. Available: www.pbs.org/wgbh/nova/teachers/activities/3413_genes_02.html. [Accès le 10 5 2019].

[10] O. A. Madeira S.C, «Biclustering Algorithms for Biological Data Analysis: A Survey,» *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 1, n° %11, pp. 24-45, 2004.

[11] T. Sergios et K. Konstantinos, Pattern Recognition, Cambridge: Academic Press, 2009.

[12] Z. QINPEI, «Cluster Validity in Clustering Methods,» University of Eastern Finland, Finland, Joensuu, 2012.

[13] S. A. Latha Mary, A. N. Sivagami et M. U. Rani, «CLUSTER VALIDITY MEASURES DYNAMIC CLUSTERING ALGORITHMS,» *ARPN Journal of Engineering and Applied Sciences* , vol. 10, n° %19, 2015.

[14] T. Andrea, «Introduction to K-means Clustering,» 6 December 2016. [En ligne]. Available: https://www.datascience.com/blog/k-means-clustering. [Accès le 10 5 2019].

[15] P. Beatriz, G. Raúl et S. A.-R. Jesús, «Biclustering on expression data: A review,» *Journal of Biomedical Informatics*, vol. 57, pp. 163-180, 2015.

[16] E. Kemal, «Application of biclustering algorithms to biological data,» The Ohio State University, Ohio, 2012.

[17] F. Adelaide, A. Wassim, E. Mourad, L. O. José et H. Jin-Kao, «A SURVEY ON BICLUSTERING OF GENE EXPRESSION DATA,» 2012.

[18] K. Sebastian, «Biclustering: Methods, Software and Application,» University of Munchen, Munchen, 2011.

[19] D. Jeffrey, N. America et O.-A. Tayo, «ICPRAM 2018 - 7th International Conference on Pattern Recognition Applications and Methods,» chez *Performance Evaluation and Enhancement of Biclustering Algorithms*, Funchal, 2018.

[20] Y. Cheng et G. M. Church, «Biclustering of Expression Data,» ISMB 200, 2000.

[21] anaconda.org, «Anaconda,» [En ligne]. Available: www.anaconda.org. [Accès le 29 6 2019].

[22] spyder, «spyder-ide.org,» spyder, [En ligne]. Available: www.spyder-ide.org. [Accès le 29 6 2019].

[23] Python Software Foundation, «Python.org,» Python Software Foundation, [En ligne]. Available: https://www.python.org/about/. [Accès le 28 06 2019].

[24] «Matplotlib.org,» [En ligne]. Available: www.matplotlib.org. [Accès le 1 7 2019].

[25] «NumPy.org,» [En ligne]. Available: www.numpy.org. [Accès le 1 7 2019].

[26] «Pandas,» [En ligne]. Available: pandas.pydata.org. [Accès le 1 7 2019].

[27] «Homepage For BiVisu,» [En ligne]. Available: www.eie.polyu.edu.hk/~nflaw/Biclustering/. [Accès le 06 07 2019].

[28] «BicSPAM,» [En ligne]. Available: http://web.ist.utl.pt/~rmch/software/bicspam/. [Accès le 06 07 2019].

[29] A. Patrikainen, «Methods for Comparing Subspace Clusterings,» Helsinki University of Technology, Seattle, 2005.

[30] «Cluster validation in unsupervised machine learning,» 10 5 2017. [En ligne]. Available: www.kkulma.github.io/2017-05-10-cluster-validation-in-unsupervised-machine-learning/. [Accès le 10 5 2019].

[31] J. Zekan, «Turning genes on or off,» [En ligne]. Available: www.dynamicscience.com.au/tester/solutions1/advertisinbadsci/gnsonoff.html. [Accès le 10 05 2019].

[32] E. Yafi, «Incorporating Subjectivity In Data Mining,» Jamia Hamdard, New Delhi, 2006.

[33] J. Wooley et H. Lin, Catalyzing Inquiry at the Interface of Computing and Biology, Washington (DC): National Academies Press (US), 2005.

[34] S. Ron et H. Guy, «Fall 2009 Analysis of DNA Chips and Gene Networks,» 2009.

[35] B. Pontes, R. Giraldez et J. Aguilar-Ruiz, «Biclustering on expression data: A review,» *Journal of Biomedical Informatics,* vol. 57, pp. 163-180, October 2015.

[36] T. Nishant, K. Arun, K. D. Sathish et S. B. Vijaya, «Biological Databases- Integration of Life Science Data,» *Computer Science & Systems Biology,* vol. 4, n° %15, p. 6, 2011.

[37] «Group 8 Project - Microarray,» [En ligne]. Available: www.cellbiology.med.unsw.edu.au/cellbiology/index.php/Group_8_Project_-_Microarray. [Accès le 10 05 2019].

[38] «Gene Expression and Regulation,» The University of Leicester,, [En ligne]. Available: www.le.ac.uk/projects/vgec/highereducation/topics/geneexpression-regulation. [Accès le 10 5 2019].

GLOSSARY

Array: An orderly geometric arrangement of features on a solid surface.

Telomere: is a region of repetitive nucleotide sequences at each end of a chromosome, which protects the end of the chromosome from deterioration or from fusion with neighboring chromosomes

Complementary DNA (cDNA): A DNA synthesized from a mature mRNA template in a reaction catalyzed by the enzyme reverse transcriptase and the enzyme DNA polymerase.

DNA Microarray: A type of nucleic acid-based multiplex technique involving high-density arrays of nucleic acids on glass that allows to evaluate mRNA abundance of up to tens of thousands of genes simultaneously.

Fluorescent in situ hybridization (FISH): A cytogenetic technique used to detect and localize the presence or absence of specific DNA sequences on chromosomes by using fluorescent probes that bind to only those parts of the chromosome with which they show a high degree of sequence similarity.

Microarray: A 2D array on a solid substrate (usually a glass slide or silicon thin-film cell) that assays large amounts of biological material using high-throughput screening methods.

Protein Microarray: A microarray that provides a multiplex approach to identify protein–protein interactions, the substrates of protein kinases, transcription factor protein-activation or the targets of biologically active small molecules.

Tissue microarray (TMA): A microarray that consists of paraffin blocks in which separate tissue cores are assembled in array fashion to allow multiplex histological analysis.

Holoenzyme: a biochemically active compound formed by the combination of an enzyme with a coenzyme.

Prokaryotes: is a unicellular organism that lacks a membrane-bound nucleus, mitochondria, or any other membrane-bound organelle.

Eukaryotes: are organisms whose cells have a nucleus enclosed within membranes, unlike prokaryotes.

Mining software repositories (MSR): analyzes the rich data available in software repositories to uncover interesting and actionable information about software systems and projects

Deterministic algorithm: is an algorithm which, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states. Deterministic algorithms are by far the most studied and familiar kind of algorithm, as well as one of the most practical, since they can be run on real machines efficiently.

ABSTRACT

In this research, we have proposed a new model of biclustering of microarray matrices. By applying the Kmeans to reordering data, and applying original *Cheng and Church* algorithm to get biclusters. Our proposed model improve the accuracy score and run time of algorithm. We verified this new model proposed with many variant synthetic datasets and calculate the score of three external evaluation measures: Relative Non-Intersecting Area (RNIA) and Clustering Error (CE), Campello Soft Index (CSI).

Keywords: *Cheng and Church, KMEANS, Biclustering, DNA Microarray, Gene Expression, Clustering.*

ملخص

في هذا البحث ، اقترحنا نموذج جديد لتحديد المصفوفات الفرعية التي لها ترابط ثنائي المحور في شريحة الحمض النووي الصبغي الدقيقة. عن طريق تطبيق خوارزمية الـ*Kmeans* لإعادة ترتيب مصفوفة الحمض النووي الصبغي، وتطبيق خوارزمية التحليل عنقودي الثنائي الأصلية الـ*Cheng and Church* للحصول على مصفوفات فرعية. يعمل نموذجنا المقترح على تحسين درجة الدقة ووقت تشغيل الخوارزمية. لقد تحققنا من نتائج النموذج المقترح مع العديد من مجموعات البيانات المصنعة ومتنوعة و قمنا بحساب قيمة التشابه مع ثلاثة معايير تقييم خارجية: (RNIA) و (CE) و (CSI).

الكلمات الدالة: *Cheng and Church*، *KMEANS*، تحليل عنقودي الثنائي المحاور، مصفوفة دي إن إيه دقيقة، التعبير الجيني، التحليل العنقودي.

ABSTRAIT

Dans cette recherche, nous avons proposé une nouvelle modelé pour effectuer une biclustering de puces d'ADN. En appliquant *Kmeans* pour réordonné la matrice, et en appliquant l'algorithme original *Cheng and Church* pour obtenir des biclusters. Notre modèle proposé améliore le score de précision et le temps d'exécution de l'algorithme. Nous avons vérifié ce nouvelle modèle proposé avec de nombreux synthétiques ensemble de données variantes et calculé le score de trois mesures d'évaluation externes : Zone relative d'intersection (RNIA), Erreur de Clustering (CE), indice de Campello Soft (CSI).

Mots-clés: *Cheng and Church, KMEANS, Classification double, Puce à ADN, Expression génétique, Clustering.*