Data Storage for Cluster Analysis of Microarray Gene Expression Data

Yan Yang

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

March 2004

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisisitons et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-91145-4
Our file Notre référence
ISBN: 0-612-91145-4

ABSTRACT

Data Storage for Cluster Analysis of Microarray Gene Expression Data

Yan Yang

Microarray technology has now becoming a systematical way to study the expression level of thousands of genes over thousands of conditions. The large-scale, high-throughput experimental methods require analysis and information processing to match. Cluster analysis of gene expression data is one of the most important analysis steps in microarray technology. By using statistical algorithms, the purpose of cluster analysis is to group genes or samples together according to their similarities in gene expression profiles. In this thesis, we study the cluster analysis of microarray gene expression data, and analyze the data required for the clustering process.

By realizing the MicroArray Gene Expression (MAGE) data standard, we design an object data model for cluster analysis of gene expression data, and construct a relational database schema. Furthermore, we integrate the database design to an existing open source microarray application --- BASE (BioArray Software Environment). To validate the database, we modify a Java-based open source cluster analysis application --- MeV (MultiExperiment Viewer). Therefore, we are able to take the gene expression data from BASE, to run the cluster application on MeV, and to store the generated clustering data back to the modified BASE database.

Table of Contents

Table of Figures

1 Introduction

In recent years there has been an explosion in the rate of acquisition of biomedical data. Advances in molecular genetics technologies, such as DNA microarrays allow us for the first time to obtain a "global" view of the cell. We can now routinely investigate the biological molecular state of a cell by measuring the simultaneous expression of tens of thousands of genes using DNA microarrays.

All organisms on Earth, except for viruses, consist of cells. Yeast, for example, has one cell, while humans have trillions of cells. All cells have a nucleus, and inside the nucleus there is DNA, which encodes the "program" for making future organisms. DNA has coding and non-coding segments, and coding segments, called "genes", specify the structure of proteins. Proteins are large molecules, like hemoglobin, that do the essential work in every organism. Genes make proteins in two steps: DNA is transcribed into messenger RNA (mRNA), which in turn is translated into proteins. Practically all cells in the same organism have the same genes, but these genes can be expressed differently at different times and under different conditions, such as tissue type, developmental stage, environment, and treatment. The different patterns of gene expression carefully tune biological programs. Virtually all major differences in cell state or type are correlated with changes in the expression level of many genes.

Microarrays have opened the possibility of creating gene expression profiles to represent many biological systems, processes or clinical interests. Such data sets of molecular information can be used as inputs to perform large-scale data analysis, in which various statistical and data mining techniques are applied to identify co-regulated genes, to distinguish normal and disease cell states, to explore biological pathways, and much more.

Gene expression data analysis can be viewed in two broad categories: (1) pattern recognition, which can be unsupervised (cluster analysis, class discovery) or supervised (discriminant analysis, class prediction); (2) detection of differential expression on a probe-by-probe basis. The most commonly used data analysis is cluster analysis. The goal of cluster analysis is to identify genes that show similar behaviors across a range of conditions or samples. The motivation to find such genes is driven by the assumption that genes that demonstrate similar patterns of expression share common characteristics, such as common regulatory elements, common functions, or common cellular origin. By grouping the genes into clusters that behave similarly, cluster analysis allows the investigator to browse the data in a less intimidating and chaotic atmosphere. In addition, many of the clustering methods are relatively easy to visualize, thus improving the accessibility of the biologically meaningful information that is in the data. Several statistical clustering algorithms have been applied to gene expression data, such as hierarchical clustering, *k*–means clustering, self-organizing maps, and principal component analysis, etc. In general, more than one clustering algorithm can be applied to a dataset, where the results of several methods are compared before the true result in this dataset is discovered [17]. Many times the decision will lie on the reproducibility of the cluster using various methods. This process allows us to gain confidence that the patterns we observe not only represent the statistical meaning of data, but also represent true biological phenomena, which are independent of the analysis method.

Because of the complexity of gene expression data, it is crucial for researchers to publish the data and scientific findings to public microarray database, where others are allowed to share the data, verify and possibly extend the experiment. However, the same increase or decrease of gene expression data observed by two different laboratories might actually be different, especially when they are using different experimental protocols and data-analysis methods. Without a standard, it is almost impossible to judge the validity of a result just by inspecting the expression changes or even the raw data. In view of this problem, the Microarray Gene Expression Data (MGED) Society (http://www.mged.org), an international non-profit organization to develop standards for microarray data, has recently proposed a standard checklist, called Minimum Information About a Microarray Experiment (MIAME) (http://www.mged.org/Workgroups/MIAME/miame.html) [5], and a standard MicroArray Gene Expression Markup Language (MAGE-ML) (http://www.mged.org/Workgroups/MAGE/mage.html) [30] for data exchange. The research community has embraced it and many major journals now require compliance with MIAME for any new submission. It is therefore advisable to ensure that the experimental design, implementation and data analysis comply with the MIAME standard.

To cope with large and frequent additions of data, both the laboratory IT system architecture and the database management system must be up-to-date for collecting, storing and managing the data in a safe and easily retrievable manner. Data analysis and gene annotation through the database reference are so crucial to the interpretation of the experiments. Although current microarray data management systems store the data sets required for data analysis, the results of the analysis, e.g. cluster of interested genes, are manually saved to a flat text file and usually interpreted one gene at-a-time using public genome databases or manual literature searches. Recent gene annotation tools have been developed to discover the biological themes within gene lists derived from microarray data [16]. By automatically updating the annotation from a complete list of genomic, literature and functional databases, the interested gene lists resulted from gene expression data can be kept in a systematically way. Therefore, keeping track of the lists of interested genes is crucial for interpreting the biological meaning of gene expression profiles.

In this thesis, we analyze the data requirements for cluster analysis of microarray data. By realizing the microarray standard MAGE-OM (MicroArray Gene Expression Object Model), we develop an object model and relational database schema to store the cluster structure and results, as well as related clustering software and algorithm data. With the integration of our design to an existing web-based microarray data management system – BASE (BioArray Software Environment), and the implementation of a Java-based clustering software application – MeV (Multiexperiment Viewer), we are able to store the complete lists of the clusters into the database for further comparison and annotation.

2 Background

2.1 DNA Microarray

With the rapid growth of biotechnology, thousands of genes have been discovered everyday by sequencing the genomes of model organisms. It is widely believed that genes and their products (RNA and proteins) function in a complicated way, and orchestrate the mystery of life. However, traditional methods in molecular biology generally work on a "one gene in one experiment" basis, which means that the throughput is very limited and the "whole picture" of gene function is hard to obtain. In the past decade, the emerge of the new technology, called DNA microarray, has totally changed this concept. By discovering the expression of thousands of genes under thousands of conditions, DNA microarray technology provides the systematic way to explore the genome. It performs very rapid analysis for the purposes of gene discovery, mapping, expression profiling, and polymorphism detection.

2.1.1 What is DNA microarray?

DNA microarrays are microscope slides that contain an ordered series of DNA samples. The number of ordered DNA samples can be hundred of thousands. Since the samples are arranged in an ordered way, data obtained from the microarray can be traced back to any of the samples. The typical DNA microarray contains several thousands of addressable genes [36].

There are two variants of the DNA microarray technology, in terms of the property of arrayed DNA sequence:

1. cDNA probes (500-5,000 bases long), which are enzymatically generated PCR (Polymerase Chain Reaction) products. It can be obtained directly from genomic DNA or gene databases including GeneBank and dbEST. After clone selection, amplification and purification, the probes are loaded in microtiter plates into an arraying robot and are mechanically spotted onto chemically modified glass slides. The robotic arrayers provide a reproducible and precise mathematical map from spots on the arrays to wells in the microtiter plates, and therefore to the cDNA clones and the genes that they represent.
2. Oligonucleotide (20-80-mer oligos) or peptide nucleic acid (PNA) probes, which are chemically synthesized in situ (on-chip) or by conventional synthesis followed by on-chip immobilization. This method, also called DNA chips, is developed at Affymetrix Inc (http://www.affymetrix.com), which uses photolithography and solid-phase chemistry to produce arrays containing hundreds of thousands of oligonucleotide probes packed at extremely high densities.

2.1.2 DNA microarray technology

DNA microarray technology is an extension of traditional Southern and Northern blots, colony hybridizations, and dot blots methods. Although the principles are all based on hybridization [Figure 1], which is also called base-pairing (i.e., A-T and C-G in DNA), DNA microarray uses thousands of addressable DNAs instead of only one gene at a time.

The ultimate goal of DNA microarray technology is to identify and quantify gene expression levels in different biological samples. Samples of interest (targets) are labeled and allowed to hybridize to the array (probe); after sufficient time for hybridization and following appropriate washing steps, an image of the array is acquired. The representation of gene expression level in the sample is reflected by the amount of hybridization to complementary DNAs, which are immobilized in known positions on the array.

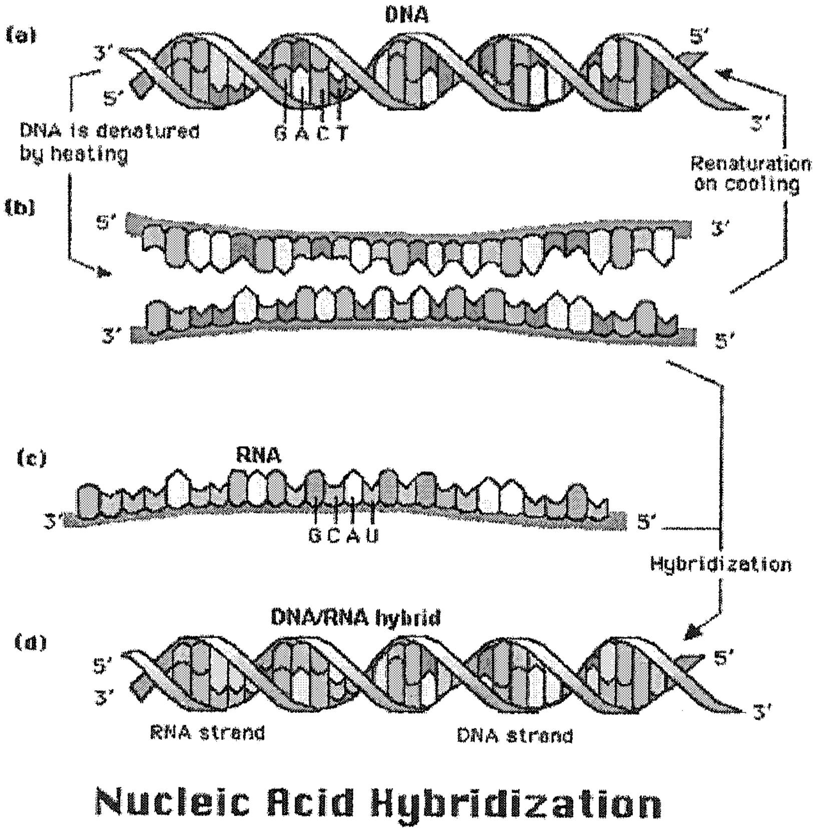


Figure 1: Process of creating a hybrid strand of DNA/RNA [21]

The two strands of a DNA molecule are **denatured** by heating to about 100°C = 212°F (a to b). At this temperature, the complementary base pairs are disrupted and the helix rapidly dissociates into two single strands. The DNA denaturation is reversible by keeping the two single stands of DNA for a prolonged period at 65°C = 149°F (b to a). This process is called **DNA renaturation** or hybridization.
Similar hybridization reactions can occur between any single stranded nucleic acid chain: DNA/DNA, RNA/RNA, DNA/RNA. If an RNA transcript is introduced during the renaturation process, the RNA competes with the coding DNA strand and forms double-stranded DNA/RNA hybrid molecule (c to d).
These hybridization reactions can be used to detect and characterize nucleotide sequences using a particular nucleotide sequence as a probe.

The step-by-step illustration of making up a comparative cDNA hybridization experiment [6] is shown on Figure 2.

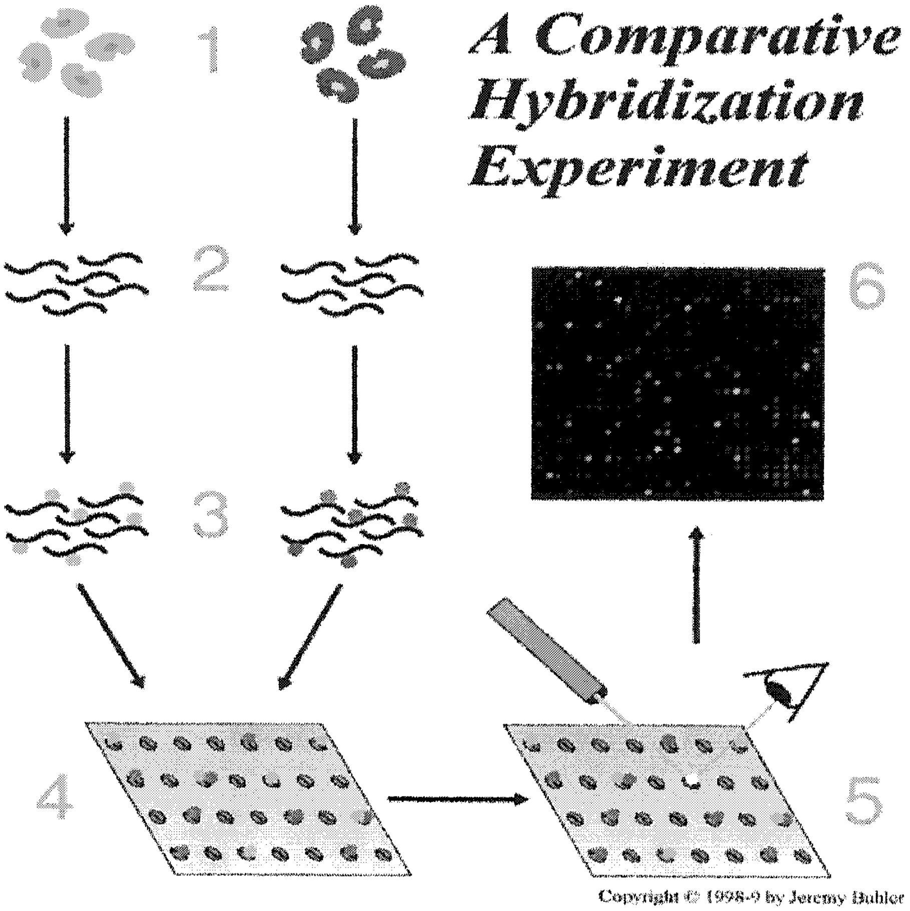

A Comparative Hybridization Experiment
1
2
3
4
5
6
Copyright © 1998-9 by Jeremy Buhler

Figure 2: A comparative cDNA hybridization experiment [6]

1. Design experiment and choose cell populations

To discover how the genes play different roles at different conditions, we have to know how genes are expressed at these conditions. However, the design issue may vary depending on individual research purposes. Typically there are four kinds of research interests at a comparative cDNA hybridization experiment:

a. Tissue specified.

Genes may express differently in liver cell and brain cell. Therefore the behaviors of the cell type are distinguished. In order to identify these genes, different tissue type cells are chosen to perform the experiment.

b. Cancer related.

Cancer can be caused by the malfunction of some regulatory genes, which normally control the growth of the cell. Cells from cancer patients and non-cancer patients are chosen for identifying these regulatory genes.

c. Cellular response to the environment.

Environment changes, for example, changes of temperature or pH, changes of nutrient availability, and the presence of environment toxins, etc, can turn some gene expressions up or down, so that the cell can respond appropriately. Choosing the cells before the change and after the change can reveal the difference of the gene expression level between them.

d. Cell cycle variation.

The cell cycle is an ordered set of events, from DNA synthesis, to mitosis (stage of producing two daughter cells), eventually to death. Many genes are involved in these activities. Experiments on cells at different cell cycle stages can help to distinguish these genes and their roles.

2. Extract mRNA and reverse transcript to cDNA

To understand how genes are expressed, Figure 3 illustrates the Central Dogma of molecular biology. DNAs are transcribed to mRNAs, which in turn are translated into proteins, which are involved in almost biological activities, structural or enzymatic.

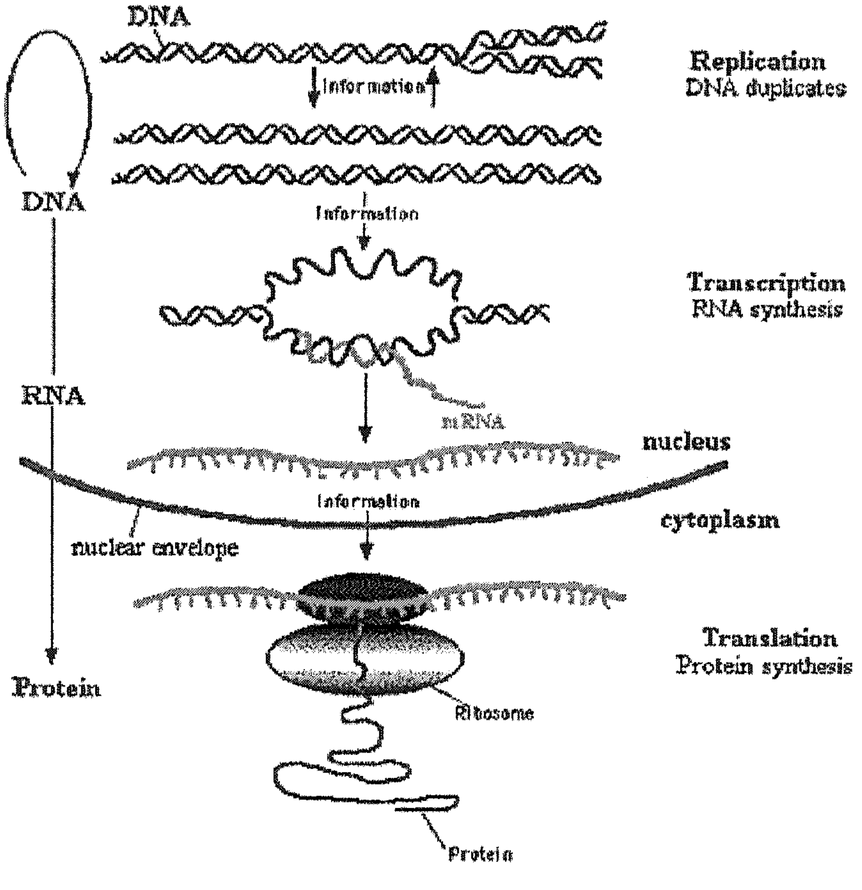


Figure 3: The Central Dogma of Molecular Biology [35]

Transcription of DNA to RNA to protein: This dogma forms the backbone of molecular biology and is represented by four major stages.

1. The DNA replicates its information in a process that involves many enzymes: **replication.**
2. The DNA codes for the production of messenger RNA (mRNA) during **transcription.**
3. In eucaryotic cells, the mRNA is **processed** (essentially by splicing) and migrates from the nucleus to the cytoplasm.
4. Messenger RNA carries coded information to ribosomes. The ribosomes "read" this information and use it for protein synthesis. This process is called **translation.**

Since proteins are technically difficult and expensive to isolate and analyze, the transcription levels of genes, which are measured by the amount of mRNAs, are used to represent the gene expression level. Those mRNAs are extracted from the cell and purified by capturing them using complementary oligodeoxythymidine (oligo(dT)) molecules bound to a solid support, such as a chromatographic column or a collection of magnetic beads. To prevent the mRNAs being destroyed by the enzymes, they are reverse transcribed into more stable DNA form, called complementary DNA (cDNA), in which there sequences are the complement to original mRNA sequences.

3. Fluorescent label of cDNA's

In order to measure the amount of cDNAs that hybridized to microarrays, cDNAs have to be labeled by fluorescent dyes. These dyes show colors under a specific frequency of light emitted by a laser. Commonly used dyes are rhodamine (Cy3) and fluorescein (Cy5), which show color red and green depending on their emission wavelength. The labeled cDNA samples are called probes because they are used to probe the collection of spots on the array.

4. Hybridize cDNA samples to a DNA microarray

Two cDNA samples are mixed and hybridized to a DNA microarray. If the sequence of a cDNA in a probe is complement to the DNA on a given spot, that cDNA will hybridize to the spot, where it will be detectable by its fluorescence. On each spot, there are enough DNAs that both probes can hybridize to it at once without interference. Therefore, every spot on an array is an independent assay for the presence of a different cDNA.

5. Scan the hybridized array

   After sufficient incubation, the hybridized array is washed to get rid of the unbounded probes. Then a laser scanner scans the array slide, and a detector, either a charge-coupled device (CCD) or a confocal microscope, captured the emitted light of each spot and recorded its intensity. Spots with more bound probes will fluoresce more intensely.

6. Interpret the scanned image

   The scanned array image is the final product of the experiment. The image color was based on the measured intensity of the spot. A spot with DNA binds predominantly to the cDNA in one cell population or the other show up as green or red, while a spot with DNA bind roughly equal amount of cDNA from each cell population show up as yellow (red + green = yellow).

2.1.3 Analyze microarray gene expression data

2.1.3.1 Types of microarray application

In recent years, microarray technology has become a standard technique used in research laboratories all over the world. Microarray application is used to monitor the expression level of genes under certain conditions, therefore to determine whether a gene is present and whether it goes up or down. With the rapid profiling of the expression levels of tens of thousands of genes at the same time, microarrays have been successfully applied to almost every aspect of biomedical research [36].

Gene expression profiling can be used to determine the function of particular genes during a particular state, such as nutrition, temperature, chemical environment, or different time point of a cell cycle. Such results could be observed as up- or down-regulation, or unchanged during particular conditions. For example, a group of genes could be up-regulated during heat shock, and as a group, these genes could be assigned as heat shock responsive genes. Some genes in this group may have already been identified as heat shock responsive, but other genes in the group may not have been assigned any function. Based on a similar response to heat shock, new functions are then assigned to the genes. Therefore, extrapolation of function based on common changes in expression remains one of the most widespread applications of microarray research. By assumption, genes that share common regulatory patterns also share the same function.

On a basic scientific level, microarrays have been used to map the cellular, regional, or tissue-specific localization of genes and their respectively encoded proteins. Microarrays have been used: at the subcellular level to map genes that encode membrane or cytosolic proteins; at the cellular level to map genes that distinguish between different types of immune cells; at the tissue region level to distinguish genes which encode hippocampus or cortex brain region specific proteins; and at the tissue level to identify genes which are expressed in muscle, liver, or heart tissues.

In Pharmacological studies, microarrays have been used to identify the genes that are regulated by a certain drug and therefore help to develop new drug targets. The guiding principle in this endeavor is that genes regulated by therapeutic agents result from the actions of the drug.

In clinical studies, microarray application is used to diagnose clinically relevant diseases. The oncology field has been especially active and to an extent successful in using microarrays to differentiate between cancer cell types. The ability to identify cancer cells based on gene expression represents a novel methodology that has real benefits. In difficult cases where a morphological or an antigen marker is not available or reliable enough to distinguish cancer cell types, gene expression profiling using microarrays can be extremely valuable [28].

A very recent application of microarrays has been to perform comparative genomic analysis. Genome projects are producing sequences on a massive level, yet there still does not exist sufficient resources to sequence every organism that seems interesting or worthy of the effort. Therefore, microarrays have been used as a shortcut to characterize the genes within an organism (structural genomics) and also to determine whether those genes are expressed in a similar way to a reference organism (functional genomics).

2.1.3.2 Microarray workflow

Microarray technology with large-scale, high-throughput experimental methods, require material and information processing systems to match. Despite the excitement generated by these technologies, exploring the massive amounts of data and interpreting them into the context of biological knowledge could be a huge challenge. Figure 4 illustrates the basic workflow of microarray technology.

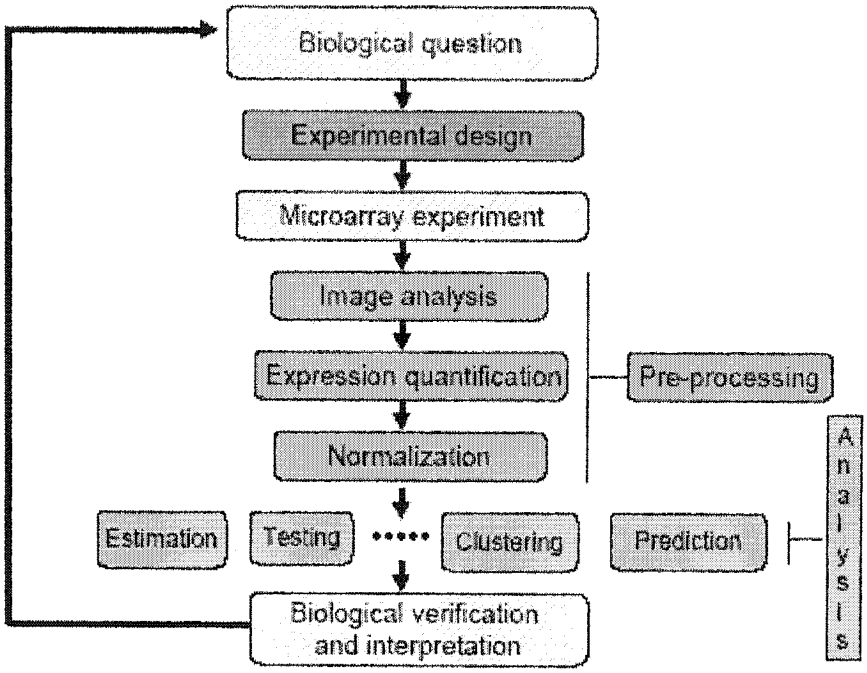


Figure 4: Microarray Workflow [22]

To solve or prove a biological question, a good experimental design is the most important part of a successful microarray experiment. In fact, the importance of pre-planning will provide with the great satisfaction and least frustration in executing a microarray project. The goal of experimental design is to remove the technical variance and to make the experiment more reliable. It may involve choosing and using appropriate controls, replicates, platforms, and statistical issues [37]. There are two main types of experimental design: the universal reference design and the dye-swap design.

Universal reference design is the most widely used experimental design for microarray. In this design, all the direct comparisons of the samples are made to a reference sample using the same orientation of dye labeling. An appropriate reference sample is the most important issue, and should be plentiful, homogenous, and stable over time. Using this design, the comparison of any two samples takes only two steps (e.g. sample A -> reference and sample B -> reference), and every new sample in the experiment is handled in the same way. This reduces the possibility of laboratory error and increases the efficiency of sample handing in large projects, which may involve large numbers of samples. However, half of the measurements in such designed experiment are made on the reference sample, and technical variation is inflated four times relative to the level that can be archived by direct comparisons [8].

Dye-swap design is a simple and effective design for the direct comparison of two samples. In this design, hybridizations on two arrays are used to compare two samples. On array 1, the control sample is assigned to the red dye, and the treatment sample is assigned to the green dye. While on assay 2, the dye assignments are reversed. The dye-swap replications can be repeated and are useful for reducing systematic bias in the red and green intensities, which require correction at the normalization step [37]. With more replicates, it is possible to obtain both an estimate of the dye effect, and a measurement of variance. Therefore, using two arrays in a dye-swap configuration to compare each sample provides technical replication and avoids confounding of effects [8]. This design is recommended for comparing a small number of sample types.

After performing a set of microarray experiments, which include microarray production, sample preparation, target labeling, hybridization and image acquisition, thousands of raw gene expression data are retrieved and produced from the microarray image. The next steps (including image analysis, expression quantification and normalization) are preprocessing steps for the raw data to produce a set of reliable gene expression data. Statistical methods and tools are needed for performing these analyses. The basic goal is to reduce an image of spots of varying intensities into a table with a measure of the intensity (or, for multi-colored fluorescence images, the ratio of intensities of each spot). Using the flags and controls, the variations due to systematic errors are removed, and data from different chips is made comparable.

Gene expression data produced by these preprocessing steps are qualified data, which have both statistically proved and quality checked values. The next step is to analyze and explore the biological meaning of the expression data. According to the knowledge of the data, analyses can include estimating, testing, clustering, classification, and prediction, etc. Statistical methods and tools played major roles in these steps. Cluster analysis is the most commonly used method to analyze the gene expression data. The goal is to identify genes that show similar patterns of expression. More details about cluster analysis will be covered in the next section.

Once the meaningful patterns and rules of large quantities of gene expression data have been discovered, we need to link the observation to the biological data, to regulation of genes, and to annotation of functions and biological processes.

2.2 Cluster Analysis

2.2.1 Overview of clustering and cluster

Clustering is the process of grouping data objects into a set of disjoint classes, called *clusters*, so that objects within a class have a high similarity to each other, while objects in separate classes are more dissimilar [15]. Clustering refers to an unsupervised learning method. *Unsupervised* means that there are no predefined classes or training examples while assigning data objects to a set of classes. Thus, clustering is distinguished from discriminant analysis (or supervised learning), which seeks to find rules for classifying objects given a set of pre-classified objects. [29]

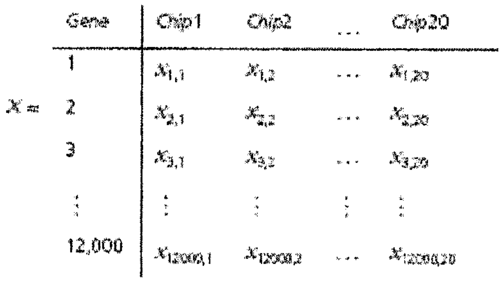
The definition of "*cluster*" is not precisely defined. In many different applications, the best definition depends on the type of data and the desired results. Several working definitions of clusters are commonly used [3]:

- Well-separated cluster definition: A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.

- Center-based cluster definition: A cluster is a set of objects such that an object in a cluster is closer to the "center" (centroid) of a cluster, than to the center of any other cluster.
- Contiguous cluster definition: A cluster is a set of points such that a point in a cluster is closer to one ore more other points in the cluster than to any point not in the cluster.
- Density-based cluster definition: A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
- Similarity-based cluster definition: A cluster is a set of objects that are "similar" and objects in other clusters are not "similar".

2.2.2 Cluster analysis of gene expression data

Preprocessed gene expression data are ready to be analyzed. The analytical goal is to find clusters of genes or clusters of samples that show similar expression patterns. Expression data are typically presented in a matrix form with each row representing a gene and each column representing a sample or chip. The matrix entry $X_{i,j}$ corresponds to the expression level of gene i in sample j.



$$X = \begin{array}{c|ccccc} \text{Gene} & \text{Chip1} & \text{Chip2} & \cdots & \text{Chip20} \\ \hline 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,20} \\ 2 & x_{2,1} & x_{2,2} & \cdots & x_{2,20} \\ 3 & x_{3,1} & x_{3,2} & \cdots & x_{3,20} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 12{,}000 & x_{12000,1} & x_{12000,2} & \cdots & x_{12000,20} \end{array}$$

The gene expression data matrix is then subjected to cluster analysis. The basic steps to the clustering process can be summarized as follows:

First, a suitable distance (similarity) between objects (based on the features) must be defined. Gene expression data objects, no matter genes or samples, can be formalized as numerical vectors $\vec{Oi} = \{o_{ij} \mid 1 \leq j \leq p\}$, where o_{ij} is the value of the *j*th feature for the *i*th data object and p is the number of features. The proximity between two objects O_i and O_j is measured by a *proximity function* of corresponding vectors $\vec{Oi}$ and $\vec{Oj}$. *Euclidean distance* is one of the most commonly-used methods to measure the distance between two data objects. The distance between objects O_i and O_j in p-dimensional space is defined as:

$$Euclidean(O_i, O_j) = \sqrt{\sum_{d=1}^{p} (o_{id} - o_{jd})^2}.$$

Other distance measurements include *Pearson's correlation coefficient, Manhattan distance* and *Spearman's rank-order correlation coefficient*, etc [9].

Second, a clustering algorithm must be selected and applied to the observed data. Generally, a clustering algorithm defines a proximity measure and a search method to find the optimal or sub-optimal partition in the data object space according to some clustering criterion. Clustering criterion is the expression of our goal of clustering which is based on a working definition of a cluster and/or an expected distribution of underlying data in the specific application domain. In the next section, we will introduce several clustering algorithms, such as Hierarchical clustering, K-means clustering, PCA, etc.

Third, cluster data results should be validated and visualized. Cluster validation is the assessment of a clustering scheme. Typically, validation indices are defined to assess the quality of clusters or to estimate the degree to which a clustering scheme fits a specific data set. Visualization of the data set is an important part of cluster analysis and is a crucial verification of the clustering results. Hierarchical clustering algorithms graphically present the results as a *dendrogram* (Figure 5) and the data set can be ordered so that the branches of the corresponding *dendrogram* do not cross and thus arranges the 'similar' data to be placed near each other.

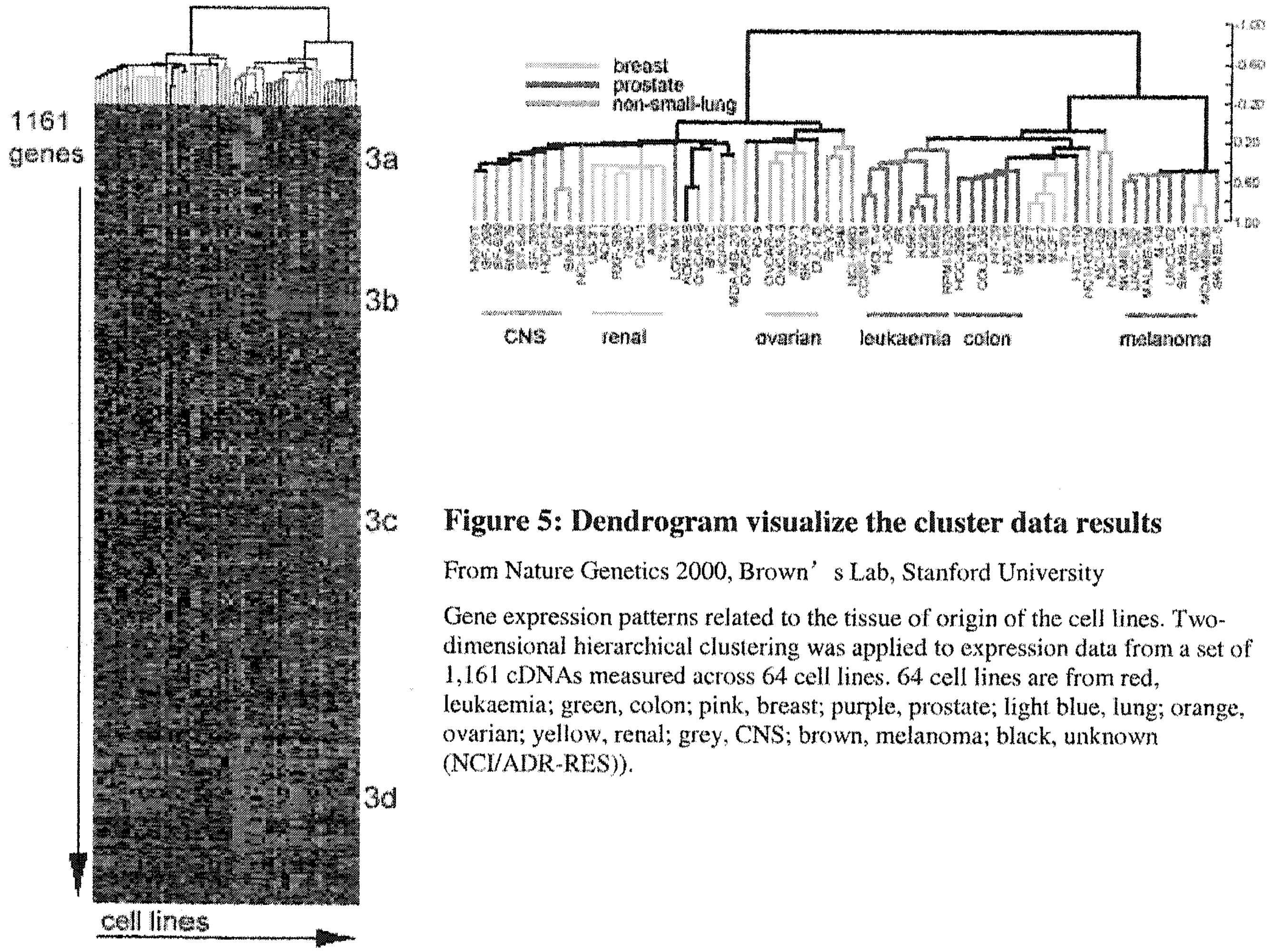


Figure 5: Dendrogram visualize the cluster data results

From Nature Genetics 2000, Brown' s Lab, Stanford University

Gene expression patterns related to the tissue of origin of the cell lines. Two-dimensional hierarchical clustering was applied to expression data from a set of 1,161 cDNAs measured across 64 cell lines. 64 cell lines are from red, leukaemia; green, colon; pink, breast; purple, prostate; light blue, lung; orange, ovarian; yellow, renal; grey, CNS; brown, melanoma; black, unknown (NCI/ADR-RES)).

Cluster analysis can apply to either genes, or samples, or both. In gene-based clustering, genes are treated as objects, while samples are considered as features. Co-related genes can be grouped together with similar expression patterns. In sample-based clustering, samples serve as data objects to be clustered, while genes play the role of features. Samples can be grouped together with some macroscopic phenotypes, such as clinical syndromes or cancer types. Both the gene-based and sample-based clustering approaches search exclusive and exhaustive partitions of objects that share the same feature space. However, another approach called "biclustering" is to capture clusters formed by a subset of genes across a subset of samples [7]. Biclustering (or subspace clustering) treats genes and samples symmetrically such that either genes or samples can be regarded as objects or features. By automatically discovering similarity based on a subset of attributes, it can simultaneous cluster genes and samples, and overlapped grouping that provides a better representation for genes with multiple functions or regulated by many factors.

2.2.3 Clustering algorithms

Hierarchical clustering

Hierarchical clustering has become one of the most widely used techniques for the analysis of gene-expression data. The advantage is that it is simple and the result can be easily visualized. Hierarchical clustering is an agglomerative approach in which single expression profiles are joined to form groups, which are further joined until the process has been carried to completion, forming a single hierarchical tree. The process of hierarchical clustering proceeds in a simple manner. First, the pairwise distance matrix is calculated for all of the genes to be clustered. Second, the distance matrix is searched for the two most similar genes or clusters; initially each cluster consists of a single gene. Third, the two selected clusters are merged to produce a new cluster that now contains at least two objects. Fourth, the distances are calculated between this new cluster and all other clusters. There is no need to calculate all distances as only those involving the new cluster have changed. Last, steps 2–4 are repeated until all objects are in one cluster [12].

One potential problem with many hierarchical clustering methods is that, as clusters grow in size, the expression vector that represents the cluster might no longer represent any of the genes in the cluster. Consequently, as clustering progresses, the actual expression patterns of the genes themselves become less relevant. Furthermore, if a bad assignment is made early in the process, it cannot be corrected.

k-means clustering

In k-means clustering, objects are partitioned into a fixed number (k) of clusters, such that the clusters are internally similar but externally dissimilar. The process is the following: First, all initial objects are randomly assigned to one of *k* clusters (where *k* is specified by the user). Second, an average expression vector is then calculated for each cluster and this is used to compute the distances between clusters. Third, using an iterative method, objects are moved between clusters and intra- and inter-cluster distances are measured with each move. Objects are allowed to remain in the new cluster only if they are closer to it than to their previous cluster. Fourth, after each move, the expression vectors for each cluster are recalculated. Last, the shuffling proceeds until moving any more objects would make the clusters more variable, increasing intra-cluster distances and decreasing inter-cluster dissimilarity [32].

Some implementations of k-means clustering allow not only the number of clusters, but also seed cases (or genes) for each cluster, to be specified. This has the potential to allow, for example, use of previous knowledge of the system to help define the cluster output.

Self-organizing maps

A self-organizing map (SOM) is a neural-network-based divisive clustering approach. A SOM assigns genes to a series of partitions on the basis of the similarity of their expression vectors to reference vectors that are defined for each partition. It is the process of defining these reference vectors that distinguishes SOM from *k*-means clustering. Before initiating the analysis, the user defines a geometric configuration for the partitions, typically a two-dimensional rectangular or hexagonal grid. Random vectors are generated for each partition, but before genes can be assigned to partitions, the vectors are first 'trained' using an iterative process that continues until convergence so that the data are most effectively separated. First, random vectors are constructed and assigned to each partition. Second, a gene is picked at random and, using a selected distance metric, the reference vector that is closest to the gene is identified. Third, the reference vector is then adjusted so that it is more similar to the vector of the assigned gene. The reference vectors that are nearby on the two-dimensional grid are also adjusted so that they are more similar to the vector of the assigned gene. Fourth, steps 2 and 3 are iterated several thousand times, decreasing the amount by which the reference vectors are adjusted and increasing the stringency used to define closeness in each step. As the process continues, the reference vectors converge to fixed values. Last, the genes are mapped to the relevant partitions depending on the reference vector to which they are most similar [18].

Principal Components Analysis

PCA is a statistical technique for determining the key variables in a multidimensional data set that explain the differences in the observations, and can be used to simplify the analysis and visualization of multidimensional data sets. It is a mathematical way that exploits the genes that have similar correlated patterns of expression, while reducing the effective dimensionality of gene-expression space without significant loss of information. Given m observations on n variables, the goal of PCA is to reduce the dimensionality of the data matrix by finding r new variables, where r is less than n. The DNA microarray data can consider the genes as variables or the experiments as variables or both. When genes are variables, the analysis creates a set of "principal gene components" that indicate the features of genes that best explain the experimental responses they produce. When experiments are the variables, the analysis creates a set of "principal experiment components" that indicate the features of the experimental conditions that best explain the gene behaviors they elicit. When both experiments and genes are analyzed together, there is a combination of these affects, the utility of which remains to be explored. Raychaudhuri *et al* [23], applied PCA to the publicly released yeast sporulation data set, confirmed that PCA can find a reduced set of variables that are useful for understanding the experiments.

Percolation clustering

Sasik *et al* [27] introduced a novel approach to the clustering of gene expression patterns in *Dictyostelium* development. Percolation clustering reveals the natural tendency of the data to cluster, in analogy to the physical phenomenon of percolation. The basic idea is to use a probe to reveal the mutual connectivity among a large number of points, with the highly-connected regions identified as clusters. Each gene expression pattern which contains m measurements, is expressed by a point in an m-dimensional space. Connecting every pair of points that are within a certain threshold distance, each connected graph that results can be regarded as a cluster. With the increase of the distance between pair of points, the points from the regions of highest density would get interconnected first to form tight clusters, next, the more dilute clusters will form. Later as connections are made between existing clusters, they merge into even larger clusters, until eventually at some large d (typically much smaller than the maximum of distance between pairs of points), all points will be interconnected.

Plaid Models

Lazzeroni *et al* [19] introduced plaid models, which allow a gene to be in more than one cluster, or in none at all. It also allows a cluster of genes to be defined with respect to only a subset of samples, not necessarily with respect to all of them. The plaid model is a form of overlapping two-sided clustering, with an embedded ANOVA in each layer. First, reorder the rows and columns in order to group together similar rows and similar columns, thus would produce an image with some number K or rectangular blocks on the diagonal. Each block would be nearly uniformly colored, and the part of the image outside of these diagonal blocks would be of a neutral background color. Second, define a layer, which describes a response that is shared by all genes in it for all samples in it. It would also be biologically interesting to identify a set of genes that had an identical response to a set of samples. Third, obtain a model that represents the data as a sum of possibly overlapping constant layers that do not have to cover the whole array. Using these models they have found interpretable structure in genetics data, foreign exchange data, and nutrition data. These structures are clearly not noise artifacts.

3 Data Analysis

In this chapter, we will analyze the data requirements for cluster analysis of microarray gene expression data, and introduce the microarray data standard MAGE-OM. An object model for cluster analysis is created using UML (Unified Modeling Language), and a relational database schema is generated accordingly.

3.1 Data requirements

3.1.1 Purpose

With the sophisticated experimental methods, and large complex data flow, microarray technology has become a cross-disciplinary endeavor requiring the collaboration of biologists, engineers, software and database designers, physicists and mathematicians. Techniques used in storing and managing the data sets can be extremely valuable to solve the biological problems [4]. Cluster analysis of gene expression data is typically the first step in data mining and knowledge discovery. The purpose of clustering gene expression data is to reveal the natural structure of the data and gain some initial insights regarding data distribution. These are following reasons to store the clustering related data:

First, cluster analysis can involve different clustering algorithms. Each algorithm may have different search criteria. Therefore, for a set of gene expression data, there are many different cluster results that may be revealed. It is important for statisticians and biologists to choose a good algorithm and suitable results from the mass quantities of data. Moreover, it is easy to explore the data and manipulate the cluster results (e.g. find the intersection of different clustering algorithms) from the database than from the flat file.

Second, interesting clusters need to be further analyzed and annotated. Data used to perform the cluster analysis are pre-processed data, which may have gone through several transformations from the raw data set. Clustering results should have connections with the original information of the genes or samples. Therefore, certain annotations (e.g. link to annotated sequence database) could be performed automatically on the clusters to reveal the biological meanings.

Third, with the development of powerful data visualization methods and tools, clustering results can produce snapshots or overviews of large expression data sets. With the clustering process information stored in the database, it is easy for researchers to review the cluster structures and images without performing clustering from the data sets.

In order to explore the large quantities of data and perform better statistical analysis, to integrate with other databases, and to visualize the clustering, it is necessary to store the data that is involved in the cluster analysis of gene expression data into the database.

3.1.2 Required data

The process of cluster analysis includes taking the pre-processed gene expression data, applying the clustering tool, and producing clustering results. Figure 6 shows the data required for cluster analysis, and details are explained below:

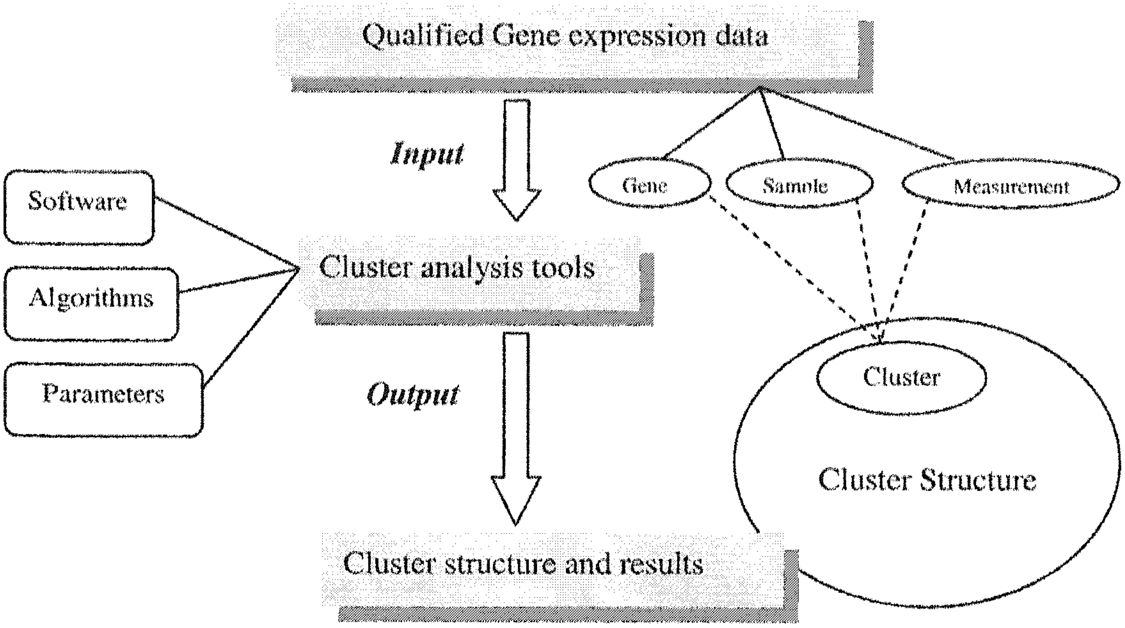


Figure 6: Data required for cluster analysis of gene expression data

Input gene expression data:

Raw gene expression data undergo several pre-processing steps, such as filtering, normalization, to generate **qualified gene expression data**, which typically is a data matrix containing the qualified **measurement** of gene expression on certain **genes** over desired **samples**.

Output cluster structure and results:

Cluster structure represents the relationship of the clusters. For different clustering of gene expression data, each cluster may contain information on either genes or samples or measurements (indicates with dashed arrow in Figure 6). Clustering results derived from most of the clustering algorithms can be represented with a tree structure. In a typical hierarchical clustering structure, the leaf node is an individual gene or sample, the internal node contains a group of genes or samples, and the root is the whole set of genes or samples.

Cluster analysis tools:

Software is the application used to perform cluster analysis. Different clustering **algorithms** can be used by different software, and **parameters** are clustering criteria used by clustering algorithms.

3.2 Microarray gene expression (MAGE) data standard

3.2.1 Standard of microarray data

Before moving on to further data analysis and data modeling, we will discuss the MAGE standard of microarray data.

Microarray data are highly context-dependent. To make sense of the data, experimental information must be provided, including what transcripts are represented, the details of the sample and any treatments, and information on other factors that may have influenced the results. Therefore, the process of publishing a microarray experiment [Figure 7] should be considered to include the steps taken to generate the data, to annotate and store the data in a local database and to transfer the data to a public repository, such as ArrayExpress of EBI (European Bioinformatics Institute) and Gene Expression Omnibus (GEO) of NCBI (National Center for Biotechnology Information) at the National Institutes of Health [31].

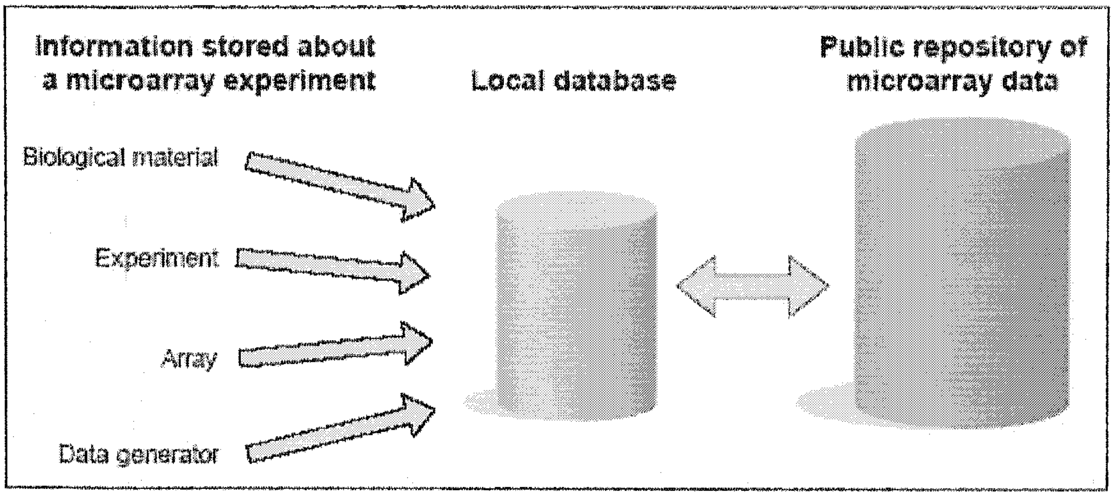

Information stored about a microarray experiment
Local database
Public repository of microarray data
Biological material
Experiment
Array
Data generator

Figure 7: Information storage and transfer of data for a microarray experiment [31]

If data from every microarray research project were stored in the same type of database with exactly the same structure, transferring data from one database to another would be a relatively straightforward proposition. But because different projects have very different needs, restrictions and resources, this is not likely to be possible. Therefore, standards for data representation and minimum information will enhance the value of an experiment, and enable researches to make comparison, as well as to understand, reanalyze and replicate microarray data.

To develop such a standard for microarray data, a group called MGED (the Microarray Gene Expression Data) society was officially founded in June 2002. MGED is an international organization of biologists, computer scientists, and data analysts that aims to facilitate the sharing of microarray data generated by functional genomics and proteomics experiments. This group established the standard for microarray data annotation and exchange, facilitating the creation of microarray databases and related software implementing these standards, and promoting the sharing of high quality, well annotated data within the life sciences community. There are three major projects developed by this group:

MIAME (Minimum Information About a Microarray Experiment) describes the information that researchers should provide to explain the procedures and biological purpose of their microarray data in adequate detail. It aims to outline the minimum information required to unambiguously interpret microarray data and to subsequently allow independent verification of this data at a later stage if required. [5]

MAGE (MicroArray Gene Expression) aims to provide a standard for the representation of microarray expression data that would facilitate the exchange of microarray information between different data systems. It has been established a data exchange model (MAGE-OM: Microarray Gene Expression – Object Model) and data exchage format (MAGE-ML: Microarray Gene Expression – Markup Language) for microarray expression experiments. MAGE-OM has been modeled using the Unified Modeling language (UML) and MAGE-ML has been implemented using XML (eXtensible Markup Language). [30]

OWG (Ontology Working Group) is assembling a large set of controlled vocabularies and ontologies that can be used to describe biological samples and their manipulations.

As of October 2002, several major scientific journals, including the Nature group, The Lancet, Cell and EMBO journal adopted MIAME recommendations as a requirement for publication of microarray experiments. That means all articles with microarray gene expression data should be MIAME compliant. At the same time, MAGE became the 'Available Specification for Gene Expression' at the OMG (Object Management Group).

The adoption of standards for microarray experiments will change the way of recording, storing and transporting data. As the data are generated, information that are described in the MIAME guidelines will be recorded, and be stored in a MIAME-compliant local research database. On publication, associated data will be exported through internet-based forms and/or MAGE-ML documents to one of the public microarray repositories. If data from outside sources are needed to complete the analysis, the researcher will download the MAGE-ML document that describes the outside the data and import it into the local research database.

Since many microarray projects started before the publication of MIAME and MAGE, in order to publish the experiments and share the data, either local databases need to be modified to meet the MIAME guideline or additional software need to be developed to map different databases to MAGE-OM. For our recent study that aims at data analysis and storage of cluster analysis of gene expression data, MAGE-OM would certainly be the first guideline for the development.

3.2.2 MAGE Object Model

MAGE-OM was developed with the purpose of capturing the objects relevant for microarray experiments, specifically for capturing the data and annotation of gene expression experiments. MAGE-OM is a framework for describing experiments done on all types of DNA microarrays, including spotted and synthesized arrays, and oligonucleotide and cDNA arrays. It is independent of the particular image analysis and data normalization methods, and allows representation of both raw and processed microarray data. Above the representation of expression measurements, it allows for comprehensive annotation of experimental results. The decision was made, however, to make the model general enough so that other technologies, such as proteomics, could potentially reuse this model [1].

MAGE-OM is a data-centric model that contains 132 classes grouped into 17 packages containing, in total, 123 attributes and 223 associations between classes. Classes in the model represent distinct things or events, and each class may have attributes as well as associations to other classes. The packages are used to organize classes that share a common purpose, for example the Array package contains classes that describe individual arrays, including detailed information on relevant manufacturing processes. The key components of MAGE-OM reflect many of the core requirements of MIAME, specifically: experiment goals and design (Experiment package); biological materials used and description of their creation (BioMaterial package); array design and purpose (ArrayDesign, BioSequence packages); array manufacture (Array package); hybridization, wash, and scan information (BioAssay package); gene-expression data (BioAssayData package). Besides, analysis on the result of an experiment is also included in HigherLevelAnalysis package. Figure 8 shows the component view of the main packages. Other utility packages support requirements shared by the above components, specifically information on people and organizations, protocols used, simple annotations, free-text descriptions, and the ability to specify links to predefined ontology [30].

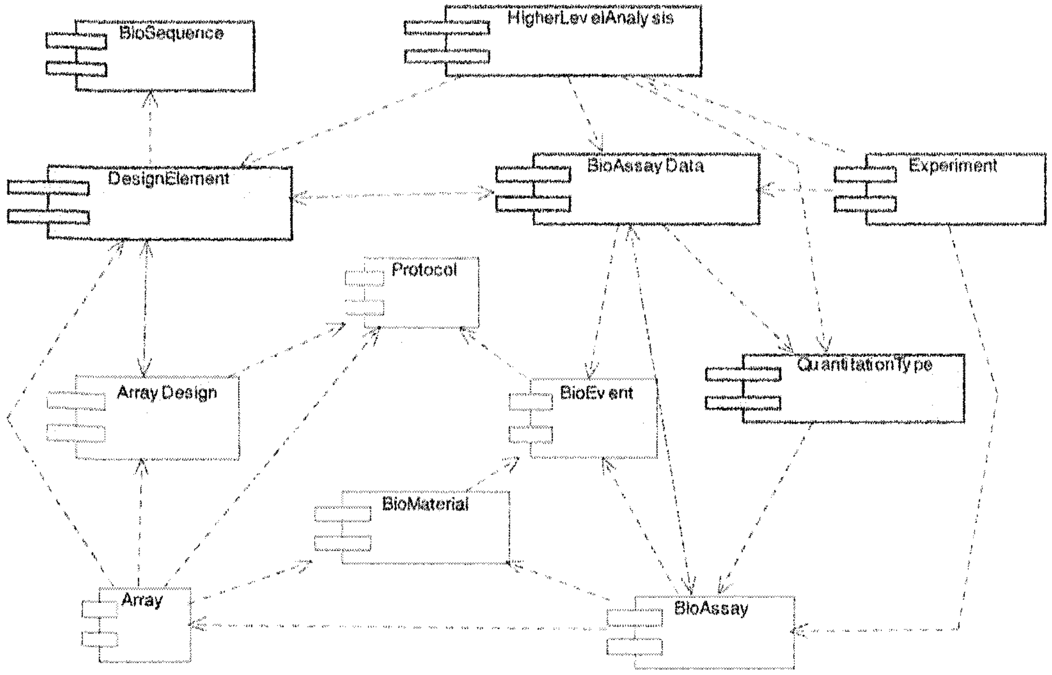


Figure 8: Component View of Main Packages of MAGE-OM

According to our data requirements for cluster analysis of gene expression data (Section 3.1), we need to explore how MAGE-OM models these requirements, in particular: cluster analysis structure and results (HigherLevelAnalysis), gene expression data (BioAssayData package), and cluster analysis tools (Protocol package).

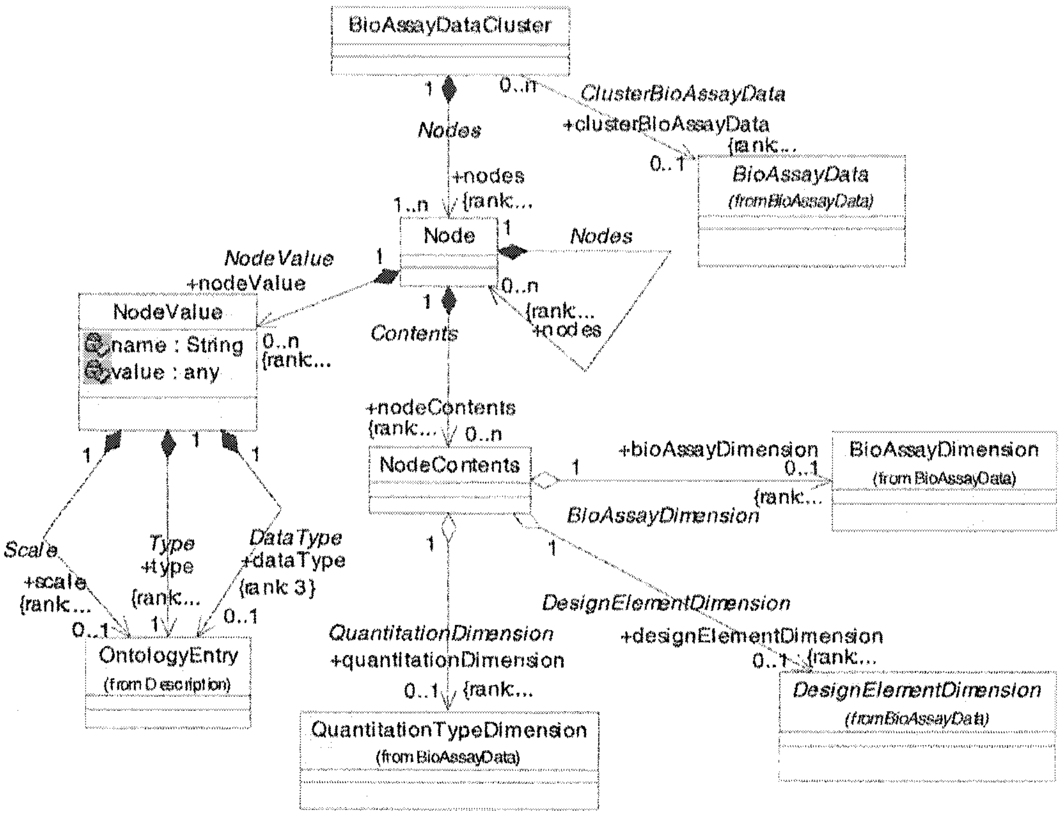


Figure 9: Class Diagram of HigherLevelAnalysis Package from MAGE-OM

Figure 9 shows the HigherLevelAnalysis Package from MAGE-OM. This package describes the results of performing analysis on the result of the BioAssayData from an Experiment. Typical examples include clustered results (from hierarchical or k-means clustering) or results from self-organizing maps. The model contains the framework supporting node- and tree-based clusters. Each clustering has an association to the data (BioAssayData) from which the results were generated, as well as one or more nodes. Each node can contain other nodes to create a tree or it can contain one or more dimensions (explained next in BioAssayData package), allowing the node to identify which of the matrix indices (e.g. genes or samples) are clustered together.

Class **BioAssayDataCluster** represents a mathematical method of higher level analysis (typically referrs to cluster analysis) whereby BioAssayData are grouped together into nodes. Class **Node** is an individual component of a clustering and may contain other nodes. Class **NodeContents** is the contents of a node for any or all of the three Dimensions. If a node only contains genes, then just the DesignElementDimension would be defined. Class **NodeValue** is a value associated with the Node that can rank it in relation to the other nodes produced by the clustering algorithm, for example, distance between parent node and child node. It has an association to OntologyEntry from which the scale (linear, log10, ln, etc), data type and the type (of value, distance, etc) are defined.

Figure 10 shows the class diagram of BioAssayData package. The classes defined in this package provide data and information, as well as annotation on the derivation of that data. **BioAssayData** is the entry point to the values of the dataset. While the actual values are represented by **BioDataValues**, BioAssayData is the source of which **Transformation** event uses to produce the DerivedBioAssayData. Recall that the data recorded from the image scanner are raw data (in MAGE-OM, denoted as MeasuredBioAssayData), it must be pre-processed through many steps, such as filtering, normalization, etc, and the data are reformatted to be DerivedBioAssayData. The process by which derivedBioAssays are created from measuredBioAssays or derivedBioAssays is called Transformation. It uses mappings to indicate the input and output dimension.

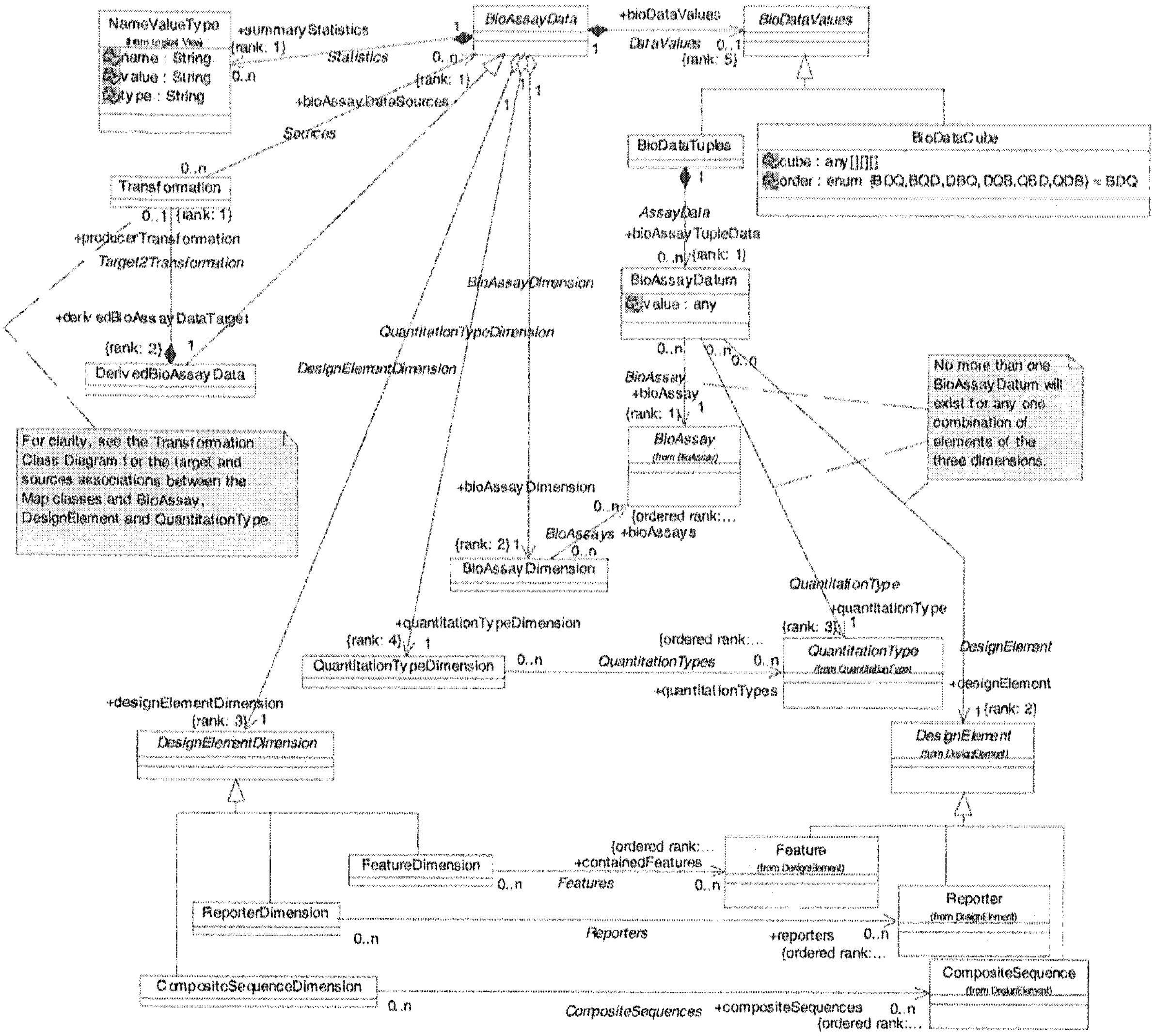
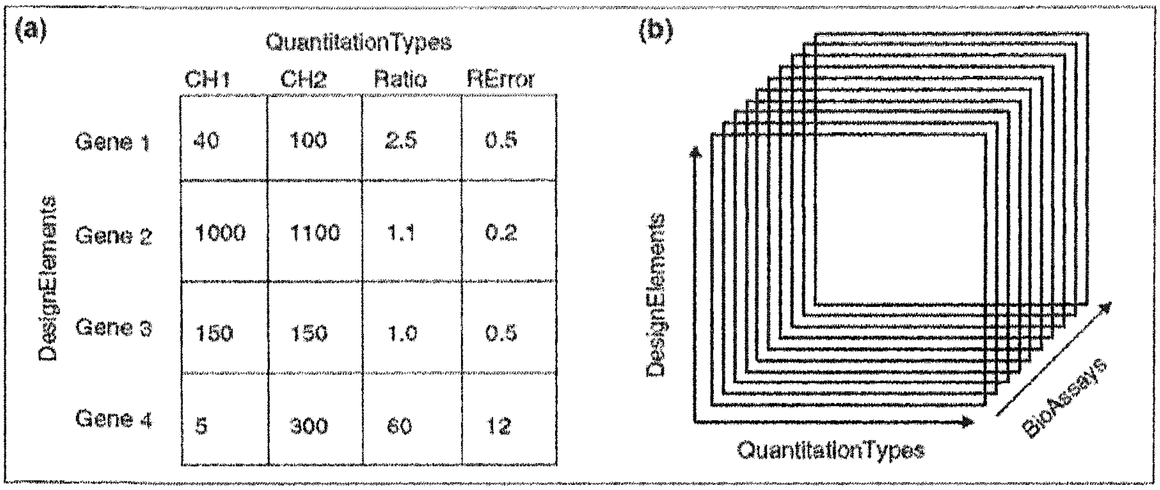


Figure 10: Class Diagram of BioAssayData Package from MAGE-OM

In MAGE-OM, the gene expression data are viewed as a three-dimensional matrix (or cube) of values [Figure 11] whose axes are labeled by DesignElements (the 'genes'), BioAssays ('experimental samples'), and QuantitationTypes (parameters from the scanning software) [30]. Therefore, BioAssayData consists of these three-dimensional data, where each dimension is an ordered list.



	CH1	CH2	Ratio	RError
Gene 1	40	100	2.5	0.5
Gene 2	1000	1100	1.1	0.2
Gene 3	150	150	1.0	0.5
Gene 4	5	300	60	12

Figure 11: The BioDataCube [30]

BioDataCubes are composed of a matrix of values. **(a)** A two-dimensional slice of a BioDataCube for a single Bioassay. Each combination of DesignElements and QuantitationTypes is allowed a value. CH1 is channel 1Foreground and Background, CH2 is channel 2 Foreground and Background, Ratio is the ratio between the background subtracted intensities of Channel 2 over Channel 1, RError is the ratio error. **(b)** The cube of values is a set of slices, in this view one slice for each BioAssay.

To summarize, BioAssayData used by cluster analysis is DerivedBioAssayData, which is from MeasuredBioAssayData through the process of Transformation. BioAssayData is annotated by a three-dimensional cube data, where DesignElementDimension represents a list of genes, BioAssayDimension is a list of samples, and QuantitationDimension is a list of measurements.

In Figure 12, Protocol package describes a generic laboratory procedure or analysis algorithm, and an instance class --- ProtocolApplication, which can describe the actual application of a protocol. The ProtocolApplication is the use of a protocol with the requisite Parameters and ParameterValues. For example, cluster analysis application is an instance of ProtocolApplication. It is performed by a Person, and uses a SoftwareApplication to execute a clustering algorithm (an instance of Protocol), with the ParameterValues of any Parameters.

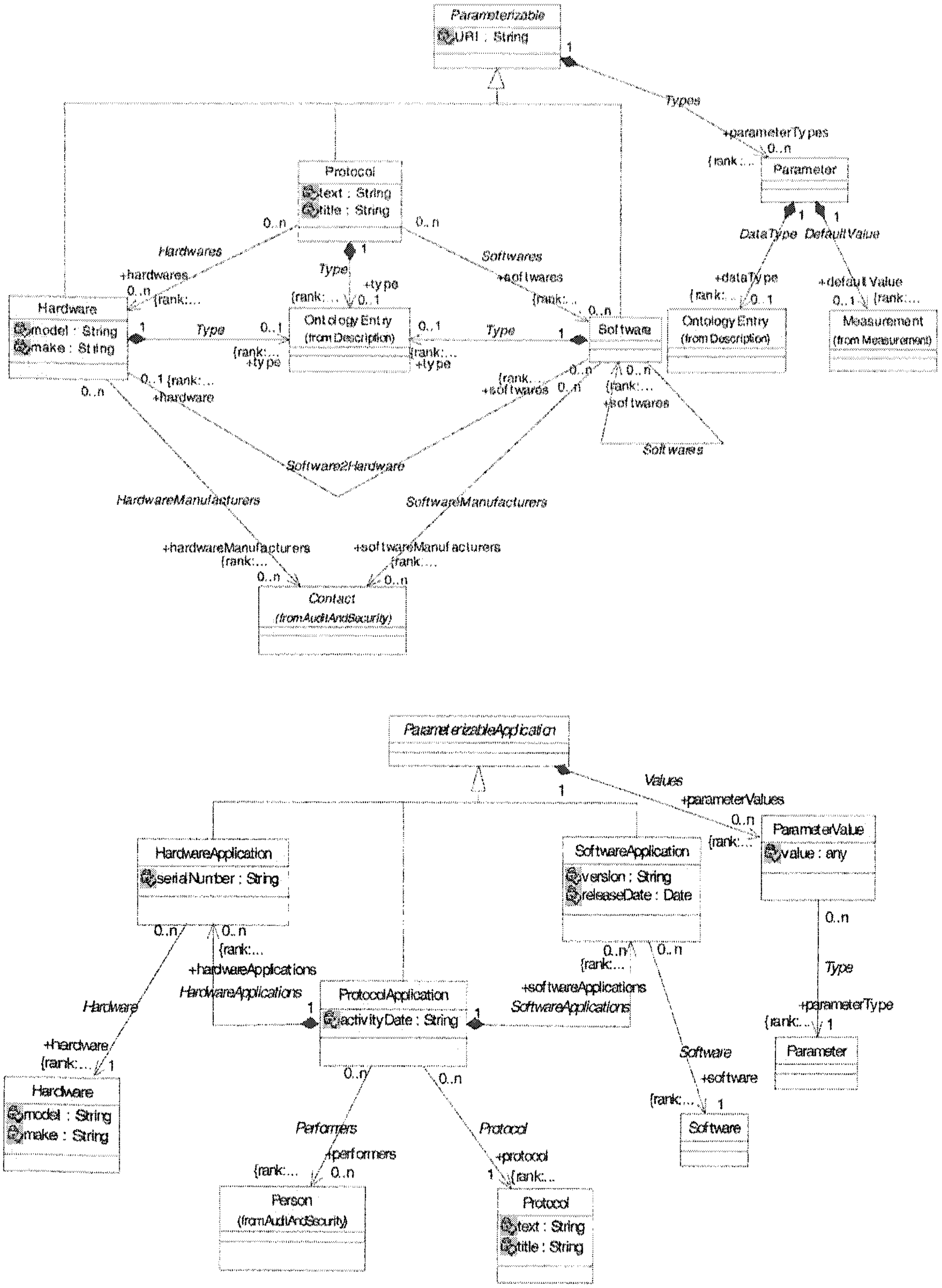


Figure 12: Class Diagram of Protocol Package from MAGE-OM

3.3 Modeling of cluster analysis of microarray gene expression data

3.3.1 Object model

By analyzing the MAGE-OM, we design our own data model for the cluster analysis of gene expression data. Figure 13 lists the modified data model to meet our data requirements.

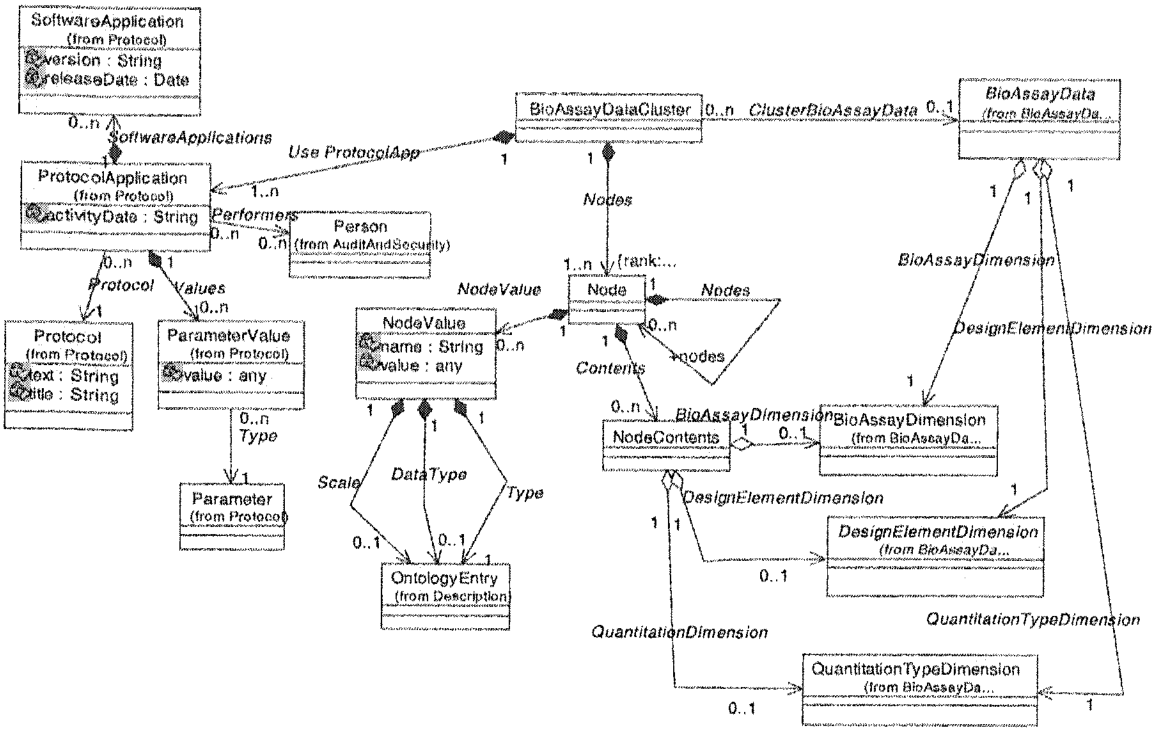


Figure 13: Object model of cluster analysis for gene expression data

Since MAGE-OM has modeled most of the scenarios for cluster structure, cluster results, and more importantly, their relationship with BioAssayData, we adopt the concepts in the HigherLevelAnalysis package and part of the BioAssayData package. However, the original BioAssayDataCluster class in HigherLevelAnalysis of MAGE-OM has no relation with the Protocol package, therefore, it is impossible to trace the clustering software application, and the clustering algorithm, as well as parameters used to perform cluster analysis. Our approach to this problem is to add an association from BioAssayDataCluster to ProtocolApplication of Protocol package, as well as related classes SoftwareApplication, Protocol, Parameter, ParameterValue, Person, and their relations to ProtocolApplication.

This conceptual model well answers our data requirements for cluster analysis of gene expression data. BioAssayData used for cluster analysis represents the qualified gene expression data, which consists of three-dimensional data. BioAssayDataCluster represents the output cluster structure Node and cluster results NodeValue and NodeContents. ProtocolApplication is used to process BioAssayDataCluster, where Protocol, SoftwareApplication and Parameter are used by ProtocolApplication.

3.3.2 Relational database schema

Based on the above object conceptual model, we built an Entity-Relation-Attribute (ERA) model of the relational database with its entire table schema described in Figure 14. The diagram was drawn by ERwin software (ERwin 4.1 by Computer Associates). The following table contains the explanation of the diagram:

□	Entity
▢	Relationship
────	Identifying relationship
- - - - - -	Non-identifying relationship
────●	One to Zero, one or many
◇────●	Zero or one to Zero, one or many
───P─●	One to one or many

The mapping from the object model to the relational model is not straightforward. It requires adding tables and attributes. The major difference is that classes in the Protocol package are changed to a cluster-specific name, e.g. ClusterApplication instead of ProtocolApplication, and the Person class used to record the person who executes the ProtocolApplication is merged as an attribute of the ClusterApplication table. In addition, NodeValueType table maps the OntologyEntry class in the object model, with the attributes of type, datatype and scale. More details are explained below:

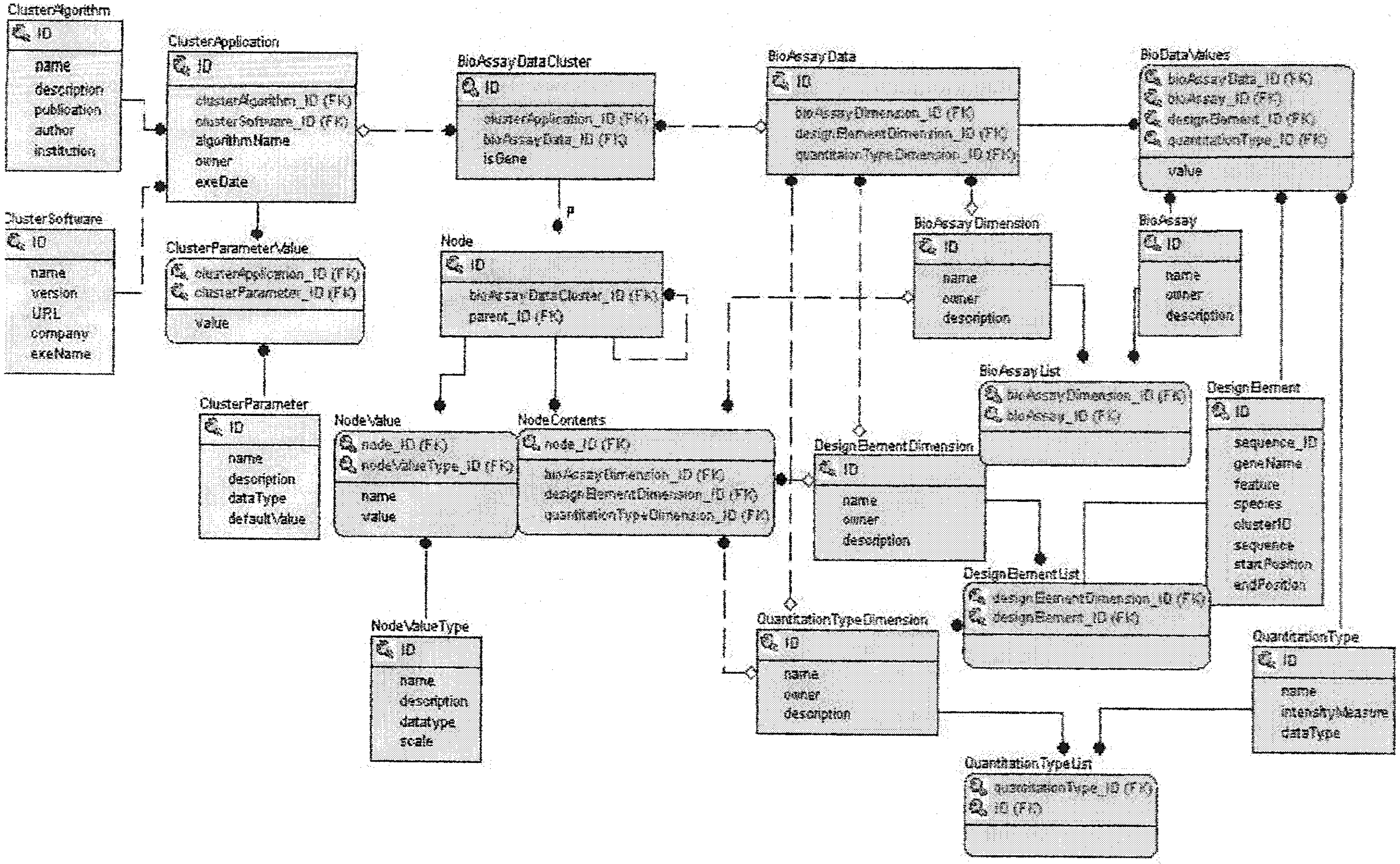


Figure 14: Relational database scheme of cluster analysis of gene expression data

According to the data requirements, there are three parts of data involved in the database:

Part I: Cluster algorithm, software, application, and parameter

1. **ClusterApplication** is used to perform a cluster analysis on gene expression data. It uses one of the **ClusterAlgorithm** in one of the **ClusterSoftware**. For example, a user selects a hierarchical clustering algorithm from Eisen lab's Cluster software, or K-means clustering algorithm from the same software, or from TM4's MeV software (TIGR).
2. **ClusterParameter** contains all the parameters used to define the **ClusterApplication**, which performs cluster analysis on either gene or assay. Parameters may include the parameters for distance measurement, and the parameters for different clustering algorithms, such as different linkage methods, similarity metric, number of clusters, and maximum cycles.

Part II: Output cluster structure and cluster results

1. **BioAssayDataCluster** uses one **ClusterApplication** to perform clustering analysis on one **BioAssayData**.
2. A **BioAssayDataCluster** can have one or many **Nodes**. Each **Node** in a tree can have only one parent **Node** (except root **Node**'s parent is NULL).
3. Each **Node** may contain some **NodeValues**, which represent the relations to other **Nodes**, such as distance. **NodeValue** is defined by **NodeValueType**. For example, node value can be the distance to the parent of the Node, or the distance to the root of the tree, etc.

4. Each Node may contain **NodeContents**, which can be either a set of BioAssay (**BioAssayDimension**), or a set of DesignElement (**DesignElementDimension**), or a set of QuantitationType (**QuatitationTypeDimension**).

Part III: Input gene expression data

1. **BioAssayData** identifies the three-dimensional data used for cluster analysis. The three dimensions are **BioAssayDimension**, **DesignElementDimension** and **QuantitationTypeDimension**.
2. **BioAssayDimension** identifies a set of **BioAssays**. One **BioAssayDimension** can have many **BioAssay**s and one **BioAssay** can be in many **BioAssayDimension**s. A relation table **BioAssayDimensionList** represents such many-to-many relations. The same relations are applied to the other two dimensions.
3. **BioDataValues** contains a set of values (usually the intensity value) in a **BioAssayData**. The value is recorded according to **BioAssay**, **DesignElement**, and **QuantitationType**.

3.3.3 Standard versus practical

Our design for cluster analysis is based on the microarray standard MAGE-OM, which is designed to accommodate MIAME compliant microarray experiments as complete and flexible as possible. Since MAGE object model encompasses many different types of information about microarray experiments, it is not expected that every user of MAGE will use all the classes, attributes or associations of the model. Instead, a given project needs only report or encode the MAGE elements that are appropriate for the data being managed. Moreover, steps involved in creating qualified gene expression data are beyond our concern. In order to get a complete picture of how the data are managed, we have to choose a microarray project to practice our design and verify its feasibility. In next chapter we will introduce microarray databases and systems, and integrate our design to an open-source microarray database and system.

4 Data Storage

In this chapter, we will introduce the current public microarray databases and applications. Upon analyzing the existing open-source systems, we apply our relational data schema of cluster analysis to the BASE (BioArray Software Environment) system.

4.1 Microarray databases and applications

4.1.1 Public microarray databases

The core function of a microarray data management system is to store, process, visualize, and compare global gene expression data. Since using a simple flat file system makes it difficult to maintain and link biological annotations that are essential for the interpretation of the data, the storage and archival capability of a microarray system should build on a relational database.

A microarray laboratory is usually run with several people participating in the common experimental workflow of array hybridization, scanning, data processing, and analysis. All members of the team will eventually need to visualize the data. Saving flat files at different locations after each step of the workflow breaks the information stream. Therefore, such a database management system should centralize the data, with an administration system that allows different users to act upon the data at different levels. Most existing microarray data management systems are designed using a three-tier architecture. The three-tiered architecture involves a database server (also known as the back-end), an application layer (the middle layer), and a graphical user interface (the front-end or GUI). The main feature of this system architecture is the centralization of the data and of the computing intensive tasks into a central machine, the server.

In addition, the inference power of high-throughput expression data systems relies on comparing as many expression profiles as possible with each other and with other sources of information. Therefore, data should be stored in a compatible format that allows such comparisons. This capability is critical to import data from different sources, to quickly add gene annotation data, and to allow complex queries of the data using standard language or controlled vocabulary. This conceptual framework has been formalized by the MIAME (Minimum Information About Microarray Experiment) definition by the MGED (Microarray Gene Expression Data) group and has currently been adopted by prominent journals, such as Nature, Science, etc. Another standard to emerge is MAGE-ML (Microarray Gene Expression Markup Language), which is a descriptive language widely adopted by several microarary database systems and applications. Both ArrayExpress from EBI (European Bioinformatics Institute) and the Gene-Expression Omnibus (GEO) from NCBI (National Center of Biotechnology Institute), which are two of the most prominent microarray data repositories, intend to support the MIAME and MAGE-ML standards. Many other data repositories are expected to follow soon.

Finally, a system that implements different protocols of data treatment, should give the investigators maximum flexibility to analyze data from different experimental designs.

The final choice of which application to adopt may rely on the current system architecture of the laboratory [2] [14].

Some of the current microarray data management systems are listed in the table below:

Name	Organization	URL
ArrayDB	National Human Genome Research Institute (NHGRI)	http://genome.nhgri.nih.gov/arraydb/
BASE	Lund University	http://base.thep.lu.se/
GeneDirector	BioDiscovery	http://www.biodiscovery.com/genedirector.asp
GeNet	Silicon Genetics	http://www.silicongenetics.com/cgi/SiG.cgi/Products/GeNet/index.smf
GeneX	National Center of Genome Research (NCGR)	http://www.ncgr.org/genex/
maxdSQL	University of Manchester	http://bioinf.man.ac.uk/microarray/maxd/maxdSQL/
NOMAD	UCSF, UCLA, Lawrence Berkeley National Laboratory	http://ucsf-nomad.sourceforge.net/help/
SMD	Stanford University	http://genome-www5.Stanford.EDU/MicroArray/SMD/

4.1.2 Open source microarray applications

Although several microarray data management systems exist to perform collecting, managing and analyzing the gene expression data, there are potential needs to develop affordable, state-of-the-art open source software with the availability of both the program source code and well-defined standards for adding functionalities and integrating them into the system.

Here we introduce three of the most widely used and comprehensive open source microarray systems [10]:

Bioconductor project: the statistical analysis tools written in R (http://www.bioconductor.org).

The main focus of Bioconductor project is to deliver high-quality infrastructure and end-user tools for expression analysis. The primary delivery vehicle is R and the R package system. The R base package and packages from the Contributed R Archive Network (CRAN; http://cran.r-project.org) provide implementations for a broad range of state-of-the-art statistical and graphical techniques, including linear and nonlinear modeling, cluster analysis, prediction, resampling, survival analysis, and time-series analysis. Additionally, R has several mechanisms that allow it to interact directly with software that has been written in many different languages (e.g., intersystem interfaces provided at http://www.omegahat.org) and allow users to incorporate additional analysis modules. Although initial efforts focused primarily on DNA microarray data analysis, many of the software tools are general and can be used broadly for the analysis of genomic and expression data. Bioconductor has adopted object-oriented programming as its primary programming paradigm. Current release of Bioconductor v. 1.3 has 51 packages grouped as General tools, Analysis, Annotation, Database interaction, Graphics & User Interface, Graphs and Pre-processing.

Bioconductor is an open development initiative. Users are encouraged to become developers, either by supplying Bioconductor-compliant packages, by adding to or improving existing packages, or by producing Bioconductor-compliant documentation. In addition to providing genomic data analysis tools, Bioconductor has a commitment to reproducible research and integrated, dynamic documentation. Each Bioconductor package contains at least one vignette, which is a document that provides a textual, task-oriented description of the package's functionality and can be used interactively. These executable documents are generated using the function Sweave from the R tools package. Additional supporting software for vignettes is being developed to aid users with obtaining data and sample code, step through specific analyses, and apply these analyses to their own data using Bioconductor's DynDoc package [10].

TM4: A Java-based system for microarray expression analysis available from The Institute for Genomic Research (TIGR; Rockville, MD, USA) (http://www.tigr.org/software) [26].

The TM4 microarray analysis suite of tools was developed to provide the microarray community with a comprehensive set of tools to handle all aspects of the microarray process. The TM4 suite of tools consist of four major applications:

- Microarray Data Manager (MADAM): a Java-based application designed to load and retrieve microarray data to and from a MIAME-compliant MySQL database (also supplied with the software). MADAM provides data entry forms, data report forms and additional applications necessary to maintain microarray data for further analysis.

- TIGR_Spotfinder: a software tool designed for Microarray image processing using the TIFF image files generated by most microarray scanners. TIGR Spotfinder was written in C/C++ for PCs running Windows NT/2000/ME/XP.
- Microarray Data Analysis System (MIDAS): a Java-based microarray data quality filtering and normalization tool that allows raw experimental data to be processed through various data normalizations, filters, and transformations via a user-designed analysis pipeline. Currently implemented normalization and data analysis algorithms include total-intensity normalization, Lowess (Locfit) normalization, flip-dye consistency checking, replicates analysis, intensity-dependent z-score filtering (slice analysis), etc.
- Multiexperiment Viewer (MeV): a Java application designed to allow the analysis of microarray data to identify patterns of gene expression and differentially expressed genes. Numerous normalization, clustering and distance algorithms have been implemented, along with a variety of graphical displays to best present the results. MeV was written to be flexible and expandable, and supports a variety of input and output formats.

TM4 is freely available to the research community and may be obtained with source code at http://www.tigr.org/software. Although these software tools were developed for spotted two-color arrays, many of the components can be easily adapted to work with single-color formats such as filter arrays and GeneChips™(Affymetrix). Three of the

TM4 applications, MADAM, MIDAS, and MeV, were developed in Java and can be run on Microsoft® Windows™, Linux®, Unix®, and MacOS X® platforms; TIGR Spotfinder was written in C/C++ and runs only on Windows systems. The TM4 software system represents a comprehensive, extensible, open-source, and freely available collection of tools that can be applied to a wide range of laboratories conducting microarray experiments.

BASE: the web-based BioArray Software Environment developed at Lund University (http://base.thep.lu.se) [25].

BASE was designed with the goal of supporting a variety of microarray platforms. Underlying the complete system is a MIAME-supportive, customizable database implemented in MySQL that tracks the elements used to construct the arrays and their annotations, the layout and design of the array itself, the biological samples used in each hybridization assay, and both the raw and transformed data; users have the option of including other LIMS components for tracking samples and reagents in the laboratory. The software that interacts with this database was developed under the Linux operating system in PHP and uses a freely available Apache Web server (http://httpdf.apache.org) to provide Web access to its functionality. The interface uses Java, JavaScript, and HTML to provide added utility, and some of the more computationally intensive analysis methods that are carried out on the server have been implemented in C++. Because of its flexible design, BASE can be used for the analysis of one- and two-color systems on a variety of substrates, cDNA and oligonucleotide arrays, Affymetrix GeneChips, and both expression analysis and comparative genomic hybridization analysis.

These three open-source microarray applications both have advantages and disadvantages [10]. Bioconductor builds on the existing power of the R statistical analysis tool development community and allows for the rapid development and dissemination of new methods. However, the R command-line environment and language complexity can be discouraging to first-time users. Several efforts are underway to simplify and enhance the user interface. TM4 gives users a graphical interface that is easy to navigate and the architecture provides great flexibility for development. However, implementation of new statistical tools requires the creation of new analysis libraries and users have to install new software releases. BASE minimizes the software update problem by using a Web-based approach and, as such, could easily integrate the Bioconductor utilities, but it loses a good deal of the graphical functionality that local applications can provide.

Among these software applications, the Fungal Genomics Project based at the Centre for Structural and Functional Genomics at Concordia University adopt BASE as the microarray software platform. In the next section, we will introduce BASE in depth, and integrate our cluster analysis data model into BASE database.

4.2 BioArray Software Environment (BASE)

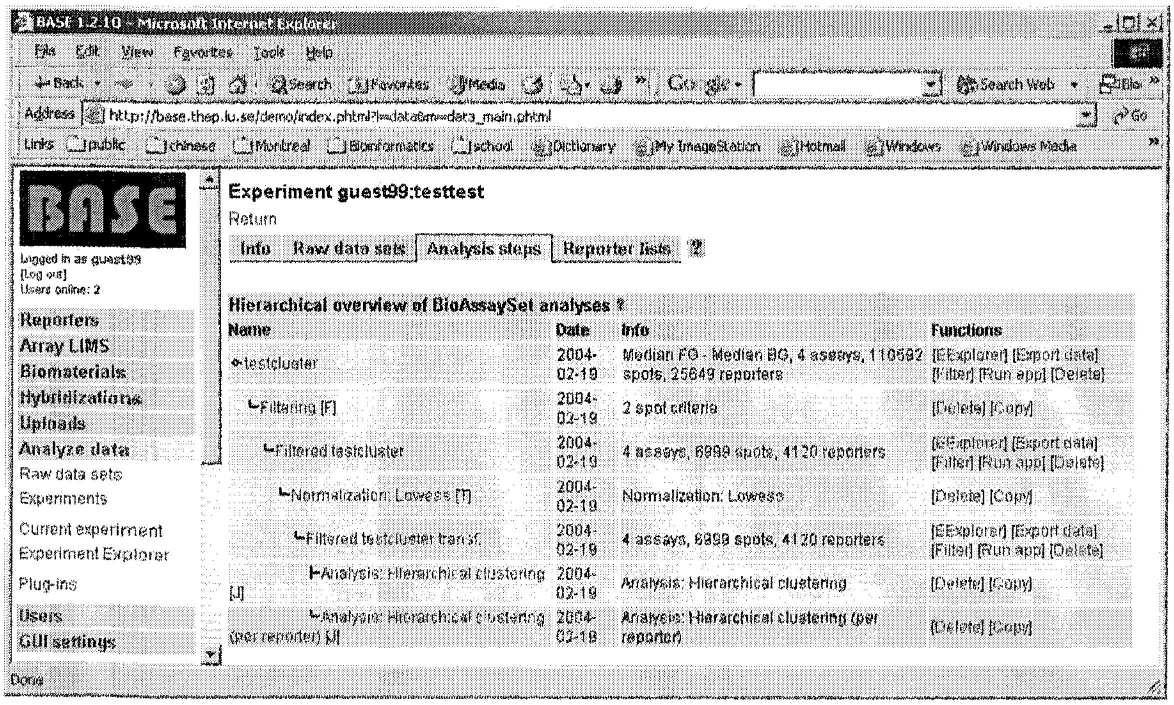
4.2.1 Overview of BASE

BASE (http://base.thep.lu.se/) is a comprehensive database server to manage the massive amounts of data generated by microarray analysis. In short, it manages biomaterial information, raw data and images, and provides integrated and "plug-in"-able normalization, data viewing and analysis tools. Additionally, for labs that make their own in-house arrays or for labs that wish to track probe information, the system also has array production LIMS features, which can be integrated with the data analysis. The organization and interface of BASE was designed to closely follow the natural workflow of the microarray biologist, and is compatible with most types of array platforms and datatypes (e.g. cDNA/oligos spotted on any substrate, Affymetrix, CGH on arrays, etc) [24].

BASE is installed on a local server in a microarray laboratory. The server is accessed via any web browser using personal login accounts with administrated access levels. With his or her own account, a user can enter data into the database, group experiments together into projects, and in a uniform and streamlined fashion, apply filters, normalizations, and run analyses. Users can choose to share almost any database item (e.g. samples, data, experiments, files, etc) with other users to facilitate online collaboration.

BASE starts the data analysis with 'Raw data set', which contains selected data from a results file. The results file is acquired when extracting numeric data from a microarray tiff image (e.g. using GenePix from Axon or ScanAlyze from Eisen's lab) and is typically a tab-delimited text file (e.g. a GenePix GPR file). Most of the data analysis in BASE is performed in the 'Experiments'. An 'Experiment' is merely a collection of 'Raw data sets' and any associated analysis steps that have been performed on these raw data sets. Raw data sets can be sorted into any number of 'Experiments' and a user can create any number of 'Experiments'. However, when analyzing data in BASE, 'BioAssays' are used as expression data rather than 'Raw data sets'. A 'BioAssay' consists of intensity values only (e.g. Int1 and int2 for a two-channel Raw data set). A 'BioAssaySet' is a collection of one or more BioAssays. 'BioAssays' are created when the 'BioAssaySet' is created. When creating a 'BioAssaySet', the quantitation of 'BioAssay' intensities should be decided, e.g. whether the intensities should be background corrected or not, how background correction should be performed and whether Mean or Median pixel values should be used for the intensities. At the same time when the 'BioAssaySet' is created, BASE will process one 'Raw data set' at the time and calculate the intensities according to the selected quantitation preferences. Each selected 'Raw data set' will be transformed into a 'BioAssay' and the corresponding 'BioAssays' will all be included in the same 'BioAssaySet' [24].

After a 'BioAssaySet' is created, a set of analysis steps include filter, normalization and cluster, as well as potential new plugins, can be performed on it. From the 'Hierarchical overview of BioAssaySet analysis' under the 'Analysis Steps' tab [Figure 15], each analysis step is clearly listed, with the link to the details of the analysis and the visualization of the data. There are four analysis functions that can be applied on 'BioAssaySet'. The 'Filter' function allows user to filter either spot or gene, and generates the 'Child BioAssaySet'. The 'Run App' function will take users to the 'Transformation:job' page where a plugin application can be selected to run. Current plugins in BASE version 1.2.10 include Multi-dimensional scaling (MDS), Standard Deviation, Normalization (Global median ratio, Lowess, pin-based Lowess), Hierarchical clustering (per sample, per reporter) and Principal Component Analysis (PCA). The 'Experiment Explorer' function is a visualization tool in which user can browse data, reporter by reporter, across all BioAssays in a BioAssaySet, including a selection of links to external databases. Users can choose to view 'LIMS info' and sample 'Annotation'. The 'Export' function faciliates the user to export the transformed data to the specific file formats used for other analysis applications, such as Eisen's Cluster software, TMEV stanford file and GeneClustering Online, as well as customized BASEfile format.


BASE 1.2.10 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media Google Search Web
Address http://base.thep.lu.se/demo/index.phtml?l=data&m=data_main.phtml Go
Links public chinese Montreal Bioinformatics school Dictionary My ImageStation Hotmail Windows Windows Media
BASE
Reporters
Array LIMS
Biomaterials
Hybridizations
Uploads
Analyze data
Raw data sets
Experiments
Current experiment
Experiment Explorer
Plug-ins
Users
GUI settings
Done

Logged in as guest99
[Log out]
Users online: 2

Experiment guest99:testtest

Return

Info | Raw data sets | Analysis steps | Reporter lists | ?

Hierarchical overview of BioAssaySet analyses ?

Name	Date	Info	Functions
testcluster	2004-02-19	Median FG - Median BG, 4 assays, 110592 spots, 25649 reporters	[EExplorer] [Export data] [Filter] [Run app] [Delete]
Filtering [F]	2004-02-19	2 spot criteria	[Delete] [Copy]
Filtered testcluster	2004-02-19	4 assays, 6999 spots, 4120 reporters	[EExplorer] [Export data] [Filter] [Run app] [Delete]
Normalization: Lowess [T]	2004-02-19	Normalization: Lowess	[Delete] [Copy]
Filtered testcluster transf.	2004-02-19	4 assays, 6999 spots, 4120 reporters	[EExplorer] [Export data] [Filter] [Run app] [Delete]
Analysis: Hierarchical clustering [J]	2004-02-19	Analysis: Hierarchical clustering	[Delete] [Copy]
Analysis: Hierarchical clustering (per reporter) [J]	2004-02-19	Analysis: Hierarchical clustering (per reporter)	[Delete] [Copy]

Figure 15: Screenshot of Data Analysis Steps in BASE

4.2.2 BASE database schema

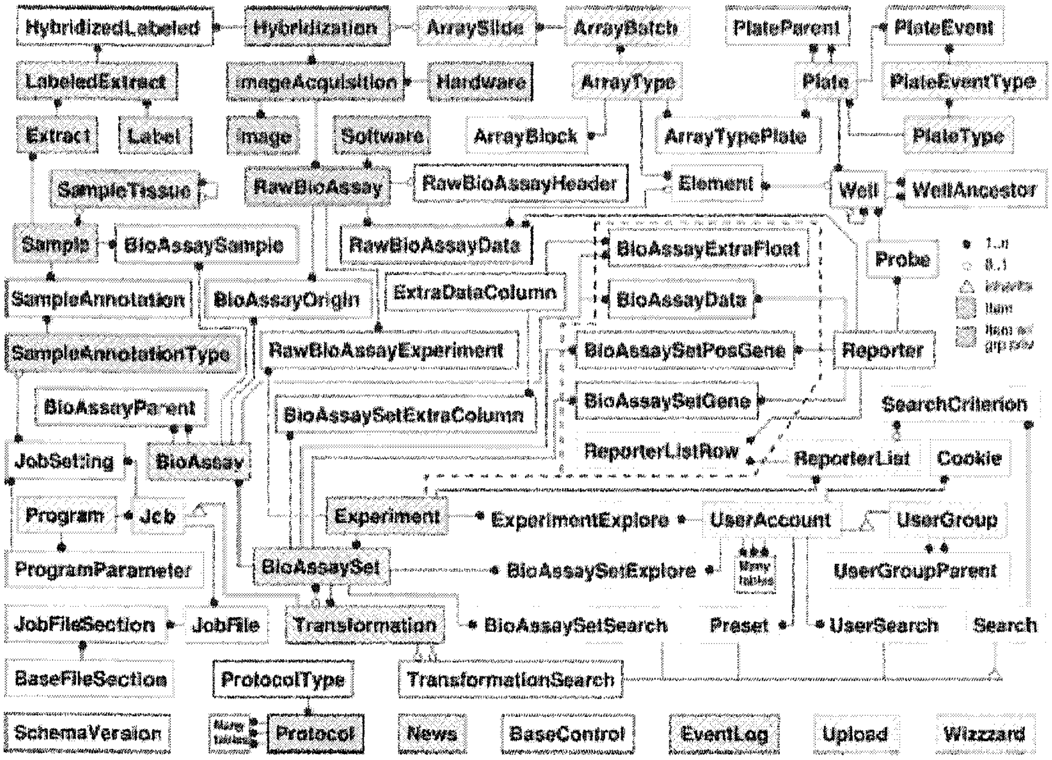


Figure 16: BASE database schema

BASE implements a relational database schema [Figure 16] with both MySQL and PostgreSQL. Although BASE does not provide a complete documentation to explain the schema, we can still trace the data through the tables and relations. Among the 76 tables in the BASE MySQL implementation, BioAssaySet, Transformation and Experiment tables are the starting points of the data analysis steps. Their relations are explained as the following:

- The Bioassaysets of an Experiment form a forest of bipartite trees, with a Transformation separating a non-root Bioassayset from its parent Bioassayset.
- A Bioassayset associated with a set of Bioassays. A Bioassay always exists as part of a single Bioassayset.

- A Bioassay consists of a number of spots. BioassayData records the intensity value of each spot, which has a unique position number in a Bioassay and a Reporter ID.
- A Transformation represents either a filtering of the data in a Bioassayset (in which case it has a single child Bioassayset), or an arbitrary transformation (in which case there may be zero or more child Bioassaysets).
- A Job is the process of Transformation. It records the execution time, status, duration, CPU time and the program name of a Transformation.

Upon analyzing the BASE schema, we find that BASE does not provide the storage for cluster analysis data, especially for cluster structure and cluster results. According to our cluster analysis database design (see Section 3.3.2), we modified BASE database schema. The next section explains the BASE-cluster database schema.

4.2.3 BASE-cluster database schema

The BASE-cluster database schema [Figure 17] is the integration of our conceptual model derived from MAGE [see Section 3.3.2] with the BASE data schema. In the diagram, the original BASE tables are shown in white, while tables shaded dark gray are added tables and shaded light gray are modified BASE tables.

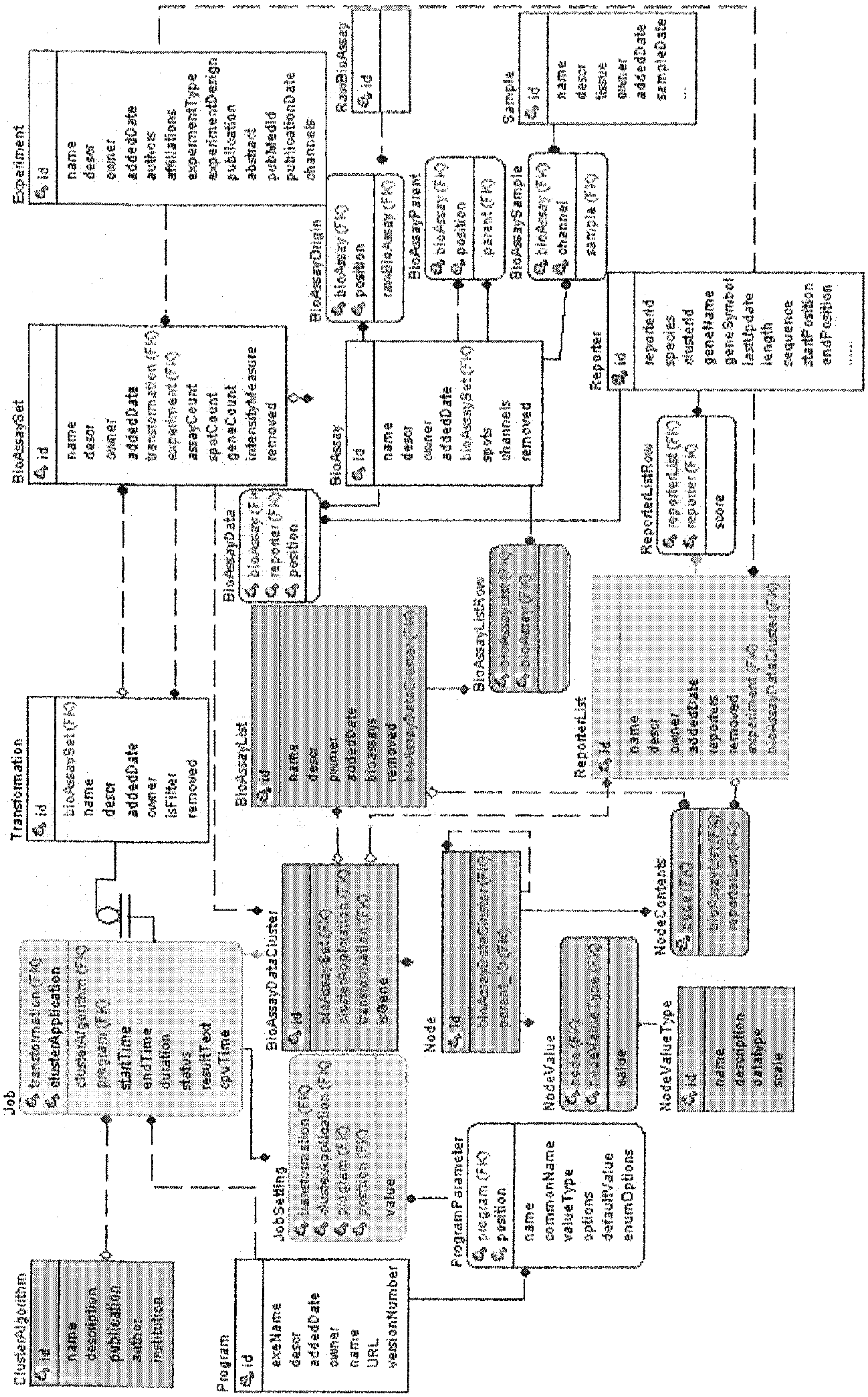


Figure 17: BASE-cluster database schema

Differences from the conceptual model are explained as the following:

1. **Transformation** stores the parent **BioAssaySet**. The root **BioAssaySet** is derived from the intensity calculation of an **Experiment**; it stores the **Experiment** and intensity measurement. A non-root **BioAssaySet** is derived by transformation of its parent **BioAssaySet**, so it records the **Transformation** number.
2. The original **Job** table is a weak-entity, which is inherited from **Transformation**. It stores the detailed process of a transformation. The modified **Job** table adds a discriminator attribute "clusterApplication" as an individual identity to identify the cluster jobs; therefore, the combined key is "transformation, clusterApplication". Also the "clusterAlgorithm" attribute is added to store the algorithm identifier used for the cluster application. Since in BASE only one algorithm is associated with a program, algorithm details are stored as the description of a program. We separated the algorithm from the program, in order to meet the situation when a program is associated with several algorithms, and one algorithm is used by many programs.
3. BASE has a **ReporterList** table to store a list of reporters (genes). Therefore, we use **ReporterList** as a DesignElementDimension of the conceptual model to store the result of gene clusters. **ReporterListRow** is the relation table that stores the reporter identifiers in a reporter list. Similarly, **BioAssayList** and **BioAssayListRow** tables are added to store the sample clusters.
4. Since the **ReporterList** table can store not only the cluster results, but also other interested lists from any files, we add a relation to **BioAssayDataCluster** and use

it to identify the cluster results from the others. This is also applied to the **BioAssayList** table.

5. We do not consider storing the QuantitationTypeDimension as part of cluster results. The reason is that BASE has its hierarchical view of data analysis, the QuantitationType (the method for calculating a single datum of a matrix) is described with the BioAssaySet and Transformation.

4.2.4 MySQL implementation

We use the BASE MySQL definition to implement the BASE-cluster database. Each entity table has the following attributes:

id: an integer of some size and the primary key of the table, has auto_increment

name: *varchar* of some size, possibly with a *UNIQUE KEY*

descr: description, text type with room for at least 64kB

owner: *int* referencing **UserAccount.**

addedDate: *date* or *datetime* describing when a record was created

removed: *tinyint* in [0, 3]

The size of *int* as an **id** is usually assigned to 11 bits. Flags and other numbers with a very limited range have been modelled as *smallint* or *tinyint*. All the tables are MyISAM types with no support on foreign key constraint. It is assumed that a column can not be *NULL*, and foreign keys are valid unless something else is explicitly mentioned. Name of the foreign key uses the same name as the referenced table.

5 Apply database to cluster analysis application

In this chapter, we will introduce three open source cluster analysis applications, and adopt MeV (Multiexperiment Viewer of TM4, TIGR) to perform cluster analysis on BASE gene expression files, while the cluster structures and results are stored at our BASE-cluster database for further comparison and annotation.

5.1 Cluster analysis application generates cluster data

For large-scale gene expression data, various statistical techniques and data mining software are used to identify significantly differentially expressed gene, understand the (dis)similarities of gene expression levels among all the samples, class prediction, and pathway analysis [20]. Cluster analysis, known as unsupervised data analysis is essentially a grouping technique that aims to find genes with similar expression profiles. Cluster analysis application can be part of a microarray system, which may includes server software, database, client software and statistics software, such as BASE (see Section 4.2); or a comprehensive software that incorporates many different analyses at different stages like data preprocessing, dimensionality reduction, normalization, clustering and visualization in a single package, but does not have any accompanied database, such as Cluster (Eisen's lab).

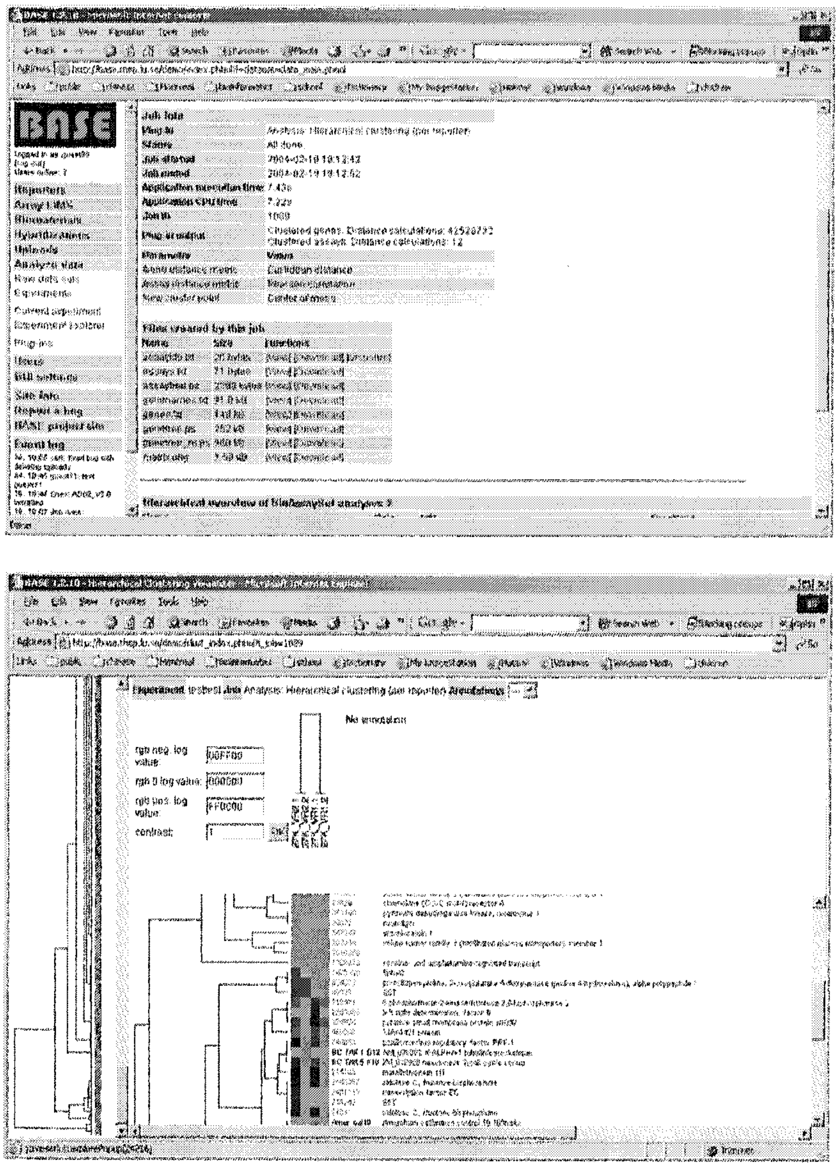
Cluster results usually can be visualized, and clusters of interests can be manually saved as a flat text file for further annotation. In order to automatically save the cluster results into our database when clustering is performed, we have to modify the cluster analysis application. A good cluster analysis application should include many available clustering algorithms, be well designed, have good visualization tools, and be consistent and reliable. In the following, we will introduce three open source cluster analysis applications, and select MeV as our cluster application to be modified.

5.1.1 Open source cluster analysis applications

BASE cluster analysis plug-ins:

Since methods for expression analysis tools are evolving rapidly, BASE has a plug-in architecture that allows new modules to be easily added for data transformation, analysis, or visualization. Any executable program that runs on Linux and can read and write a standard data format (currently their "BASEfile" format) can be adapted as a plug-in. In BASE current release 1.2.10, there are two cluster analysis plug-ins: hierarchical clustering and PCA (Principal Component Analysis).

A PhD student, Cecilia Ritz, at BASE group developed hierarchical a clustering plug-in. The hierarchy structure is built from bottom-up with two closest points are merged and the new cluster is represented by an unweighted (median) or weighted (center of mass) average of the two points in gene expression space. This algorithm takes the transformed data set in BASEfile format, and generates the text files for the cluster results. The interactive cluster tree view can be visualized through the web browser, though the picture loading time can be slow [Figure 18].



Job info

Plug-in	Analysis: Hierarchical clustering (per reporter)
Status	All done
Job started	2004-02-19 18:12:42
Job ended	2004-02-19 18:12:52
Application execution time	7.43s
Application CPU time	7.22s
Job ID	1069
Plug-in output	Clustered genes. Distance calculations: [illegible] Clustered assays. Distance calculations: 12
Parameter	Value
Gene distance metric	Euclidean distance
Assay distance metric	Pearson correlation
New cluster point	Center of mass

Files created by this job

Name	Size	Functions
[illegible]	[illegible]	[View] [Download] [illegible]
[illegible]	21 bytes	[View] [Download]
[illegible]	[illegible]	[View] [Download]
[illegible]	[illegible]	[View] [Download]
[illegible]	[illegible]	[View] [Download]
[illegible]	[illegible]	[View] [Download]
[illegible]	[illegible]	[View] [Download]
[illegible]	[illegible]	[View] [Download]

Hierarchical overview of BioAssaySet analyses ?

Experiment: testest Job Analysis: Hierarchical clustering (per reporter) Annotations

No annotation

rgb neg. log value: 00FF00

rgb 0 log value: 000000

rgb pos. log value: FF0000

contrast: 1 OK

Figure 18: Screenshot of Hierarchical clustering plugin of BASE

The PCA program is developed by Gregory O. Voronin and Ronald P. Hart in Neuroscience Gene Expression Laboratory at Rutgers University, Piscataway, NJ (http://www.ngelab.org/). This program calculates the PCA of the sample variance-covariance matrix of the data set, which is parsed from a lowess transformed data set in serial BASE file format. The variance-covariance matrix and the eigensystem (which is stored in a linked list) are used to generate the output files, currently: covariance.html, eigensystem.html, Scree.png and CoeffieicentPlot.png.

Since BASE is a web-based system, users do not need to install the program locally. In data analysis, user simply takes the pre-processed expression file and chooses a clustering plug-in, the program will be executed at the server side, with the visualization images being displayed in the client web browser and the output files are ready to download or view through the browser. However, the visualization image size can be very big, and will take some time to be loaded to the browser. The quality of the image is not good enough and the clusters of interest are not easy to be saved. Both hierarchical clustering and PCA were written in the C language. However, they were developed by different people at different institution, with no common structures to share.

With the underlying BASE database, it is straightforward to use its own programs to populate the cluster structure and results into the database for further annotation. However, there are only two separated developed clustering programs, which are not object-oriented with no further compatible clustering methods can be added in. Therefore, we do not consider implementing these applications to store the cluster related data into the database.

Cluster and TreeView:

Cluster and TreeView are developed by Michael Eisen's lab at the Lawrence Berkeley National Lab (LBNL) and the University of California at Berkeley (UCB) [13]. The programs provide a computational and graphical environment for analyzing data from DNA microarray experiments, or other genomic datasets. The Cluster program performs a variety of types of cluster analysis, including hierarchical clustering, self-organizing maps (SOMs), k-means clustering and principal component analysis. TreeView allows the organized data to be visualized and browsed [11].

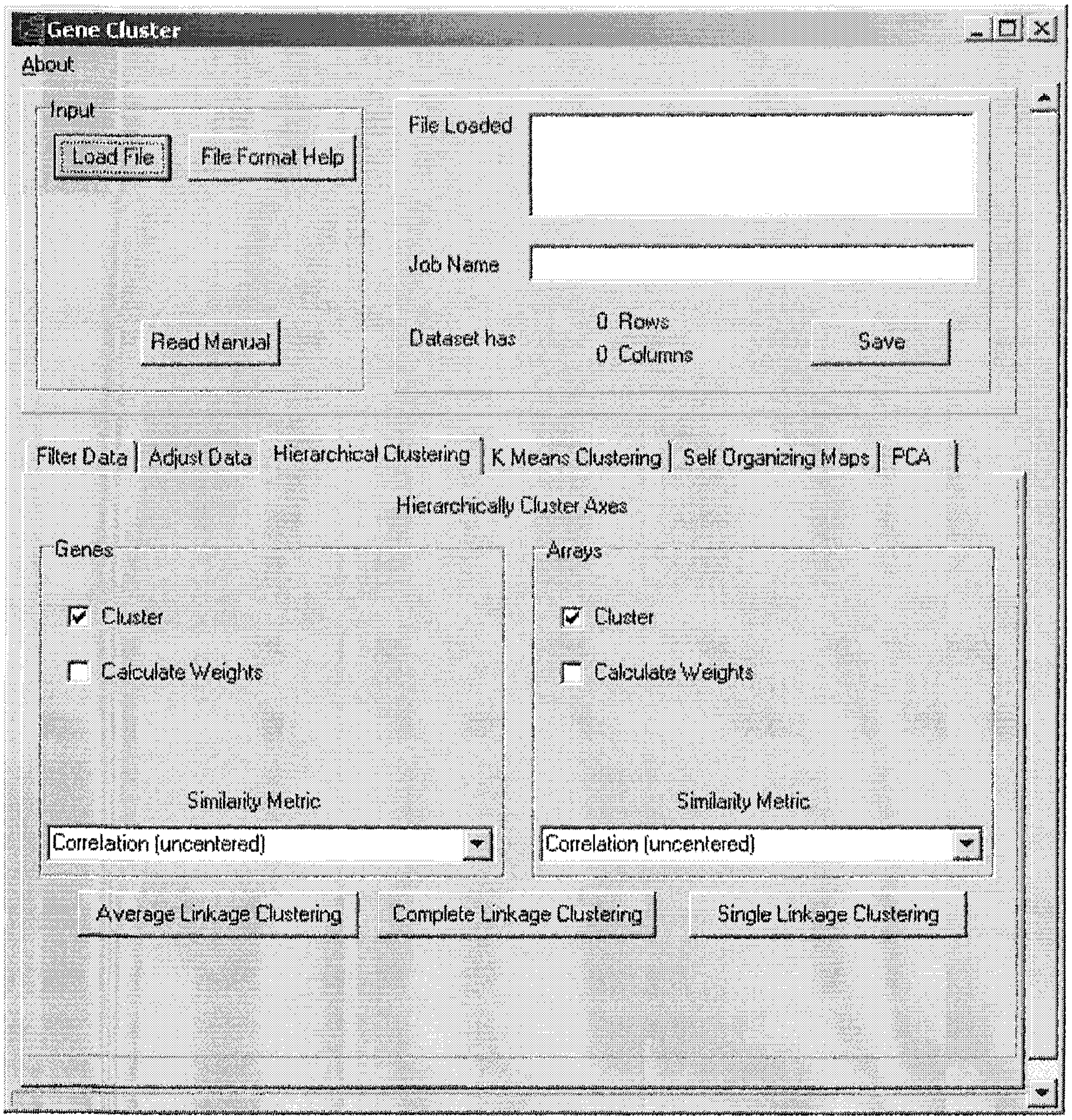
Cluster reads tab-delimited text files in a particular format, and also provides a number of options for adjusting and filtering the data. For the cluster results, Cluster writes up to three output files for each hierarchical clustering run. The three output files are JobName.cdt, JobName.gtr, JobName.atr.

The .cdt (for clustered data table) file contains the original data with the rows and columns reordered based on the clustering result. The .gtr (gene tree) and .atr (array tree) files are tab-delimited text files that report on the history of node joining in the gene or array clustering (note that these files are produced only when clustering is performed on the corresponding axis). When clustering begins each item to be clustered is assigned a unique identifier (e.g. GENE1X or ARRY42X). These identifiers are added to the .cdt file. As each node is generated, it receives a unique identifier as well; starting is NODE1X, NODE2X, etc… Each joining event is stored in the .gtr or .atr file as a row with the node identifier, the identifiers of the two joined elements, and the similarity score for the two joined elements. These files look like:

NODE1X	GENE1X	GENE4X	0.98
NODE2X	GENE5X	GENE2X	0.80
NODE3X	NODE1X	GENE3X	0.72
NODE4X	NODE2X	NODE3X	0.60

TreeView visualization program simply reads the above files, and generates an interactive graphic analysis diagram. Upon selecting a tree node, user can view the node content, e.g. gene IDs and names, as well as save the data and image. Unfortunately, only the result from hierarchical clustering can be viewed through TreeView, it does not support the visualization of other clustering, such as K-means, PCA and SOM.

Cluster and TreeView are developed by Michael Eisen at 1998, current Cluster version is 2.20 (updated at December 2002) and TreeView version is 1.6 (updated at November 2002). The programs are relatively small (about 5M each) and can be easily installed, however, it only runs on Windows environment (see Figure 19). It was written in Borland C++ with source code be freely downloaded from Eisen's lab (http://rana.lbl.gov/EisenSoftware.htm).



File Loaded

Job Name

Dataset has 0 Rows 0 Columns

Hierarchically Cluster Axes

Genes

☑ Cluster

□ Calculate Weights

Similarity Metric: Correlation (uncentered)

Arrays

☑ Cluster

□ Calculate Weights

Similarity Metric: Correlation (uncentered)

Figure 19: Screenshot of Eisen lab's Cluster software

Since their simplicities and full functionalities, Eisen lab's Cluster and TreeView software are widely spread through many microarray laboratories. However, with the emerging of many new clustering algorithms, it lacks the flexibility to add new modules to perform analysis and visualization. Moreover, the input files are restricted to a certain format, with no support for other files, such as Affymetrix file. Most importantly, it runs only on Windows, while most of the genomics projects are running at Linux or Unix. Therefore, we have to move on to search for another cluster analysis application to meet our requirements.

MeV:

TIGR MultiExperiment Viewer (MeV) is one member of a suite of microarray data management and analysis applications developed at The Institute for Genomic Research (TIGR). Within the suite, known as TM4, there are four programs: MADAM, Spotfinder, MIDAS and MeV (see section 4.1.2). Together, they provide functions for managing microarray experimental conditions and data, converting scanned slide images into numerical data, normalizing the data and finally analyzing that normalized data. MeV is an application that allows the viewing of processed microarray slide representations and the identification of genes and expression patterns of interest. Slides can be viewed one at a time in detail or in groups for comparison purposes. A variety of normalization algorithms and clustering analyses allow the user flexibility in creating meaningful views of the expression data.

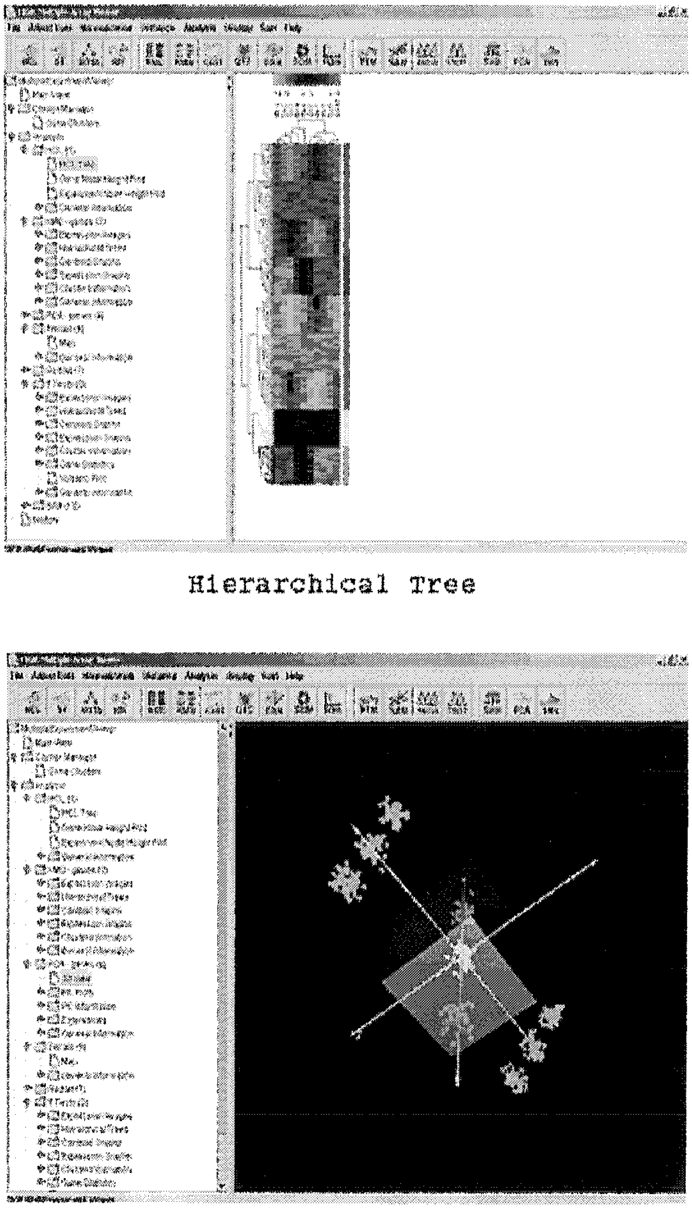
MeV can interpret files of several types, including the MultiExperiment Viewer format (.mev), the TIGR ArrayViewer format (.tav), the Stanford file format, the Affymetrix file format, and GenePix file format (.gpr). Prior to starting an analysis, certain data adjustments include normalization for experiments (where the experiments are normalized with each other), log transformations, and various filters, can be performed [33].

There are 14 clustering modules included in current release of MeV version 2.2 (http://www.tigr.org/software/tm4/mev.html):

	Module	Reference
1	Hierarchical clustering	Eisen, M.B., P.T. Spellman, P.O. Brown, and D. Botstein. 1998. Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. USA 95:14863-14868.
2	k-means clustering	Soukas, A., P. Cohen, N.D. Socci, and J.M. Friedman. 2000. Leptin-specific patterns of gene expression in white adipose tissue. Genes Dev. 14:963-980.
3	Self-organizing maps	Kohonen, T. 1992. Self-organized formation of topologically correct feature maps. Biol. Cybernetics 43:59-69. Tamayo, P., D. Slonim, J. Masirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E.S. Lander, and T.R. Golub 1999. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. Proceedings of the National Academy of Sciences USA 96:2907-2912.
4	Principal components analysis	Raychaudhuri, S., J.M. Stuart, and R.B. Altman. 2000. Principal components analysis to summarize microarray experiments: application to sporulation time series. Pac. Symp. Biocomput. 455-466.
5	Cluster affinity search technique	Ben-Dor, A., R. Shamir, and Z. Yakhini. 1999. Clustering gene expression patterns. J. Comput. Biol. 6:281-297.
6	Template matching	Pavlidis, P., and W.S. Noble 2001. Analysis of strain and regional variation in gene expression in mouse brain. Genome Biology 2:research0042.1-0042.15.
7	T-test	Pan, W. 2002. A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. Bioinformatics 18: 546-554. Dudoit, S., Y.H. Yang, M.J. Callow, and T. Speed 2000. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. Technical Report 2000, Statistics Dept., Univ. of California, Berkeley.
8	Significance Analysis of Microarrays (SAM)	Tusher, V.G., R. Tibshirani and G. Chu. 2001. Significance analysis of microarrays applied to the ionizing radiation response. Proceedings of the National Academy of Sciences USA 98: 5116-5121.
9	QT_Clust	Heyer, L.J., S. Kruglyak, and S. Yooseph. 1999. Exploring expression data: identification and analysis of coexpressed genes. Genome Res. 9:1106-1115.
10	Support Vector machines	Brown, M.P., W.N. Grundy, D. Lin, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares, Jr., and D. Haussler. 2000. Knowledge-based analysis of microarray gene expression data by using support vector machines. Proc. Natl. Acad. Sci. USA 97:262-267.
11	Gene shaving	Hastie, T., R. Tibshirani, M.B. Eisen, A. Alizadeh, R. Levy, L. Staudt, W.C. Chan, D. Botstein, and P. Brown. 2000. 'Gene shaving?as a method for identifying distinct sets of genes with similar expression patterns. Genome Biol.

		1:RESEARCH0003.
12	Relevance networks	Butte, A.J., P. Tamayo, D. Slonim, T.R. Golub, and I.S. Kohane. 2000. Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. Proc. Natl. Acad. Sci. USA 97:12182-12186.
13	Self Organizing Trees (SOTA)	Dopazo J., J. M. Carazo 1997. Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree. J. Mol. Evol. 44:226-233. Herrero, J., A. Valencia, and J. Dopazo 2001. A hierarchical unsupervised growing neural network for clustering gene expression patterns. Bioinformatics 17(2):126-136.
14	Figures of Merit (FOM)	Yeung, K.Y., D.R. Haynor, and W.L. Ruzzo 2001. Validating clustering for gene expression data. Bioinformatics 17:309-318.

Each module has its own graphic display to present the results of the module calculation (see Figure 20). Under the main navigation tree, each module has its result tree, where user can navigate through the cluster results and can even load a new session to perform sub-clustering on the cluster of interest. Clusters of interest can be stored to a repository that is managed by Cluster Manager under the main navigation tree. Gene clusters and experiment clusters are maintained in separate spreadsheets, which are viewable from the Cluster Manager node. When storing a cluster, the cluster name and a description of the algorithm or interesting features of the cluster, along with a user defined color, are stored for reference and further operation, which include launch new session for sub-clustering, and perform Union, Intersection and XOR on two or more selected clusters. Cluster data can be saved to a tab-delimited text file with the original row/column data, log ratio expression values, and (optionally) Cy3 and Cy5 values of each gene in the cluster are recorded. The expression matrix also can be saved as a tab-delimited text file, which reflects any data adjustments that are currently imposed on the data set such as percentage cutoffs or low intensity cutoffs [33].



Hierarchical Tree

Principal Components for Microarrays

Figure 20: Screenshot of MeV
(http://www.tigr.org/software/tm4/mevScreenshots.html)

The latest version of MeV 2.2 was released at July 2003. The source code and complete documentation can be obtained at TM4 online (http://www.tigr.org/software/tm4/download.html). The program is object-oriented and written in Java language, which is platform independent and can run on Windows, Linux and Unix with Java Runtime Environment (JRE) 1.3 or later installed. As newly developed software, MeV contains 14 clustering algorithms with well-defined modules to be visualized and compared. The program is consistent and flexible. Therefore, we consider adding more functionality to it, such as store the cluster results to our database.

5.1.2 Choose MeV for implementation

After running and testing three open source cluster analysis applications, we decided to connect our BASE-cluster database to MeV. The main purpose is to validate the database and to populate sample cluster structure and cluster results to the database.

The MeV framework has three layers (see Figure 21): the bottom layer – Algorithms implementation layer provides the statistical calculations and the infrastructure of the Algorithms data, the middle layer – Cluster View implementation layer constructs the sophisticated Graphic User Interface (GUI) views which are used to visualize the results of the cluster calculation, and the top layer – TIGR Multiple View layer serves as the gateway to load data and display the window's interface. There are two main APIs (Application Program Interface) [34] to connect these three layers. Algorithm API is a hierarchy of Java classes, which serve as the interface to a sophisticated calculation algorithm. This API is designed with the flexibility to create an algorithm that can use all the standard Java types and some helpful types to store arrays of float values. Each algorithm must implement these interfaces to create a concrete algorithm. Upon calculation, results are stored for cluster view to display. Cluster view API serves as the interface to GUI views, which are used to enter algorithm parameters, to execute the program, and to display progress and result of the calculation. For each algorithm, there is a corresponding GUI implementation to display the cluster result, either dendrogram or the lists of clusters, or network, 3-D structure, etc.

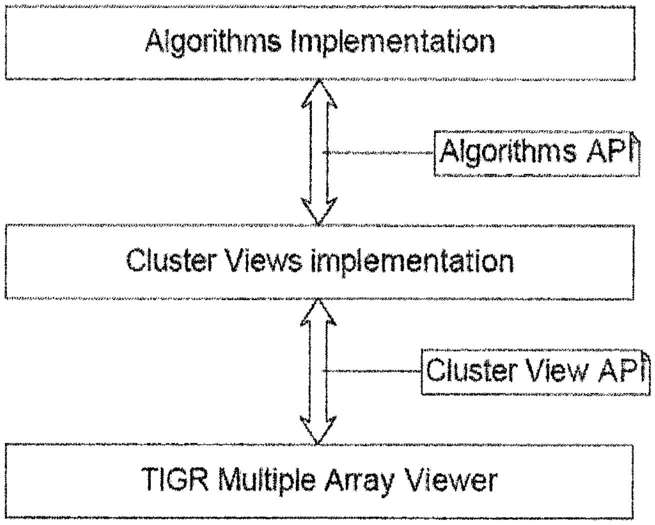

Algorithms Implementation
Algorithms API
Cluster Views implementation
Cluster View API
TIGR Multiple Array Viewer

Figure 21: Application Layer of MeV

MeV has built a complete infrastructure for executing and visualizing the cluster algorithm. It well separates the graphic view from the algorithm model. The cluster results are well handled by AlgorithmData class, and can be stored in matrix, cluster, or the integer array. Other utility classes, such as ClusterRepository and ExperimentUtility, are used to save the cluster and perform cluster operations. A sequence diagram of cluster algorithm execution is shown on Figure 22.

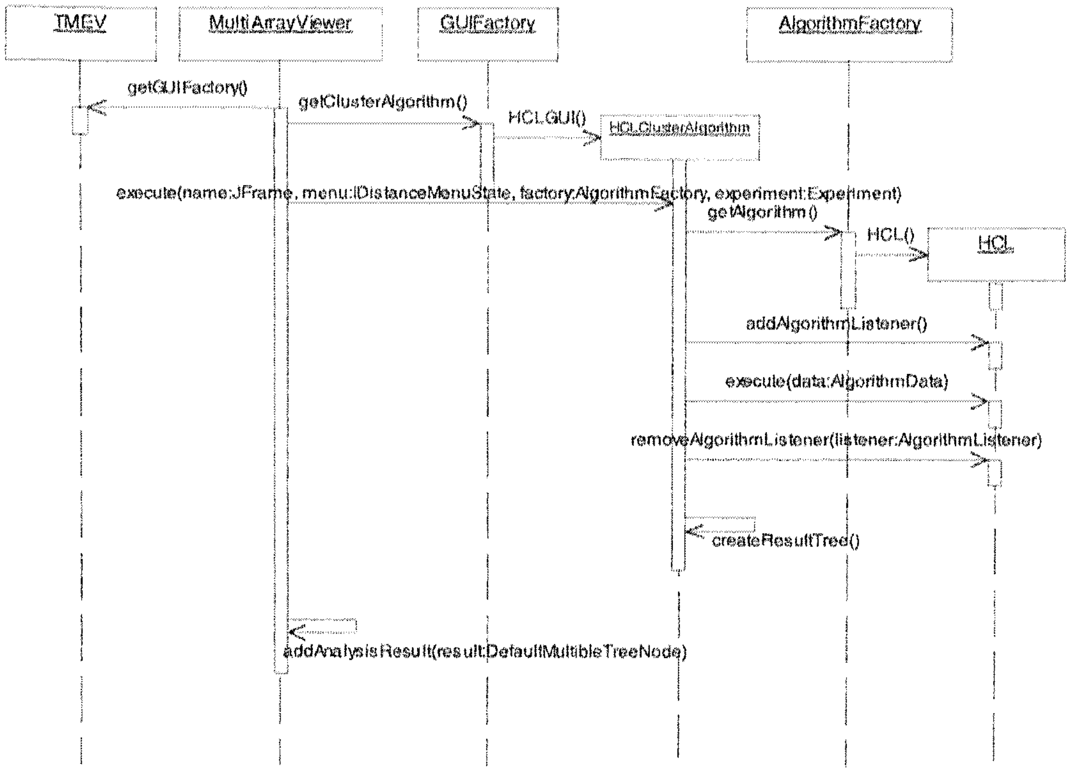

TMEV
MultiArrayViewer
GUIFactory
AlgorithmFactory
getGUIFactory()
getClusterAlgorithm()
HCLGUI()
HCLClusterAlgorithm
execute(name:JFrame, menu:IDistanceMenuState, factory:AlgorithmFactory, experiment:Experiment)
getAlgorithm()
HCL()
HCL
addAlgorithmListener()
execute(data:AlgorithmData)
removeAlgorithmListener(listener:AlgorithmListener)
createResultTree()
addAnalysisResult(result:DefaultMultibleTreeNode)

Figure 22: Sequence diagram of cluster algorithm execution

5.2 Steps to generate cluster data

Since our purpose is to verify the BASE-cluster database, after understanding the structure of MeV implementation, we choose the most commonly used algorithm --- Hierarchical clustering algorithm to generate cluster data and store to the database. The following sections explain the steps involved in the testing process.

5.2.1 Load gene expression data

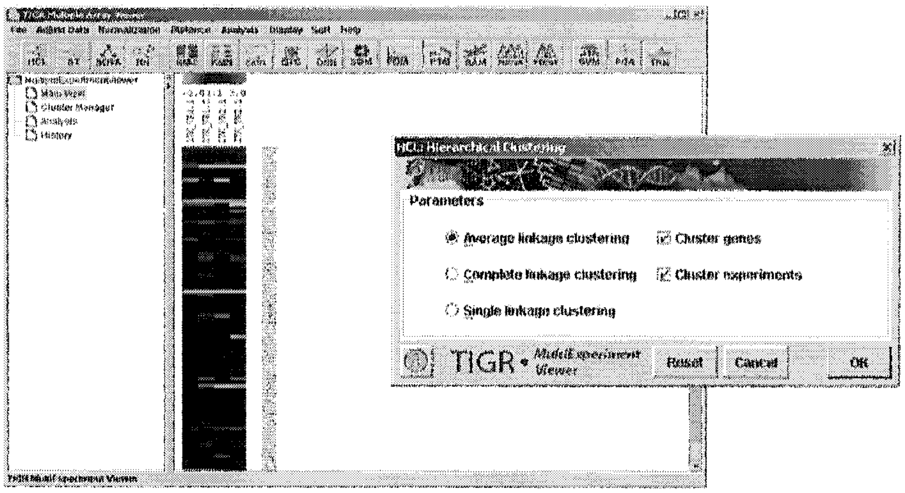
At data analysis steps of BASE, gene expression data are called BioAssaySet. After several transformations, the qualified BioAssaySet data are ready for cluster analysis. At this moment, we use the "Export data" function to export the data in MeV Stanford file format showed on the table below. The file is a tab-delimited plain text file. It contains the reporter ID, reporter name, and the intensity value for each sample.

YORF	NAME	GWEIGHT	27K_VR1.1	27K_VR1.2	27K_VR2.1	27K_VR2.2
EWEIGHT			1	1	1	1
753234	zinc finger protein, X-linked	1	0.155709	0.346859	-0.15546	-0.45347
71626	Multiple unigene's : Hs.183291 & Hs.425991	1	0.578823	-0.05514	0.546282	-0.08058
50794	zinc finger protein 133 (clone pHZ-13)	1	0.330502	0.718671	0.226952	0.410886
768644	POM (POM121 homolog, rat) and ZP3 fusion	1	-0.63735	0.005571	-0.77386	-0.0951

At the multiple array viewer window of MeV, this Stanford formatted file can be easily loaded and the data can be passed to the different algorithms.

5.2.2 Choose clustering algorithm

Upon clicking the HCL (Hierarchy clustering) icon on the algorithm bar, a HCL parameter window will be displayed for user to select the linkage method, e.g. average, complete or single, and choose either genes or experiments, or both to clustering [Figure 23].


TIGR Multiple Array Viewer
File Adjust Data Normalization Distance Analysis Display Sort Help
HCL ST SOTA RN KMC KMS CAST QTC GSH SOM FOM PTM SAM ANOVA TTEST SVM PCA TRN
MultipleExperimentViewer
Main View
Cluster Manager
Analysis
History
TIGR MultiExperiment Viewer
HCL: Hierarchical Clustering
TIGR • MultiExperiment Viewer
Reset
Cancel
OK

Parameters

● Average linkage clustering ☑ Cluster genes

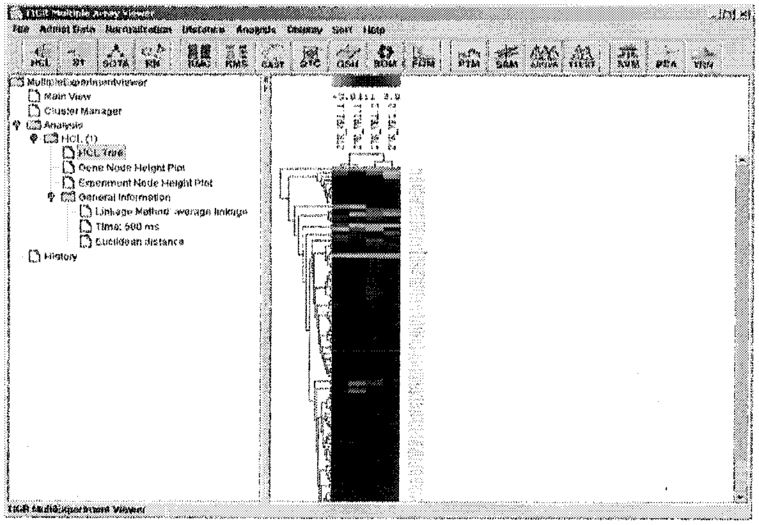
○ Complete linkage clustering ☑ Cluster experiments

○ Single linkage clustering

Figure 23: Screenshot of choosing a cluster algorithm in MeV

5.2.3 Generate and store the cluster structure and data

The dendrogram is generated by MeV after the hierarchical clustering algorithm has been performed on the original gene expression data [Figure 24].


TIGR Multiple Array Viewer
File Adjust Data Normalization Distance Analysis Display Sort Help
HCL ST SOTA RN KMC KMS CAST QTC GSH SOM FOM PTM SAM ANOVA TTEST SVM PCA TRN
MultipleExperimentViewer
Main View
Cluster Manager
Analysis
HCL (1)
HCL Tree
Gene Node Height Plot
Experiment Node Height Plot
General Information
History
TIGR MultiExperiment Viewer

Linkage Method: average linkage

Time: 500 ms

Euclidean distance

Figure 24: Screenshot of hierarchical cluster tree view in MeV

At the same time, the cluster tree structure is stored to the Node table of our BASE-cluster database. For each node, an auto-incremented identification number is assigned, and its parent identification number is stored and pointed to the same table, along with a bioAssayCluster number which identifies this particular tree. In the original implementation of MeV, the node and tree structure are stored in three integer arrays: child_1_array, child_2_array and node_order. Each node points to its two children, and node order decides the index of the node. This design is easy for displaying the tree; however, it is different from our database design in which the node is pointed to its parent. We modified the structure of MeV and successfully stored the designed data structure to the database.

5.2.4 Link to original microarray data

The next step is to store the node contents and link to the original microarray data. In BASE, the original tables are Reporter and BioAssay. We use recursive programming to get the lists of genes or samples and store to the ReporterListRow table and BioAssayListRow table. The identification number of the reporter or bioassay can be linked to the original Reporter or BioAssay tables. Therefore, a complete annotation of the list of genes or bioassays can be easily obtained.

From the above steps, we successfully populate the cluster structure and data to the BASE-cluster database. However, these steps are only for testing purpose, for complete implementation, we need to modify every algorithms in MeV. By understanding the class structure and how the algorithm data are stored and manipulated, it is not difficult to finish the whole implementation in the near future.

6 Conclusion and future work

6.1 Conclusion

Microarray data analysis is developing rapidly with new and more complex methods and systems appearing everyday. The ability to store, query and compare gene expression data has become crucial to the progress of scientific research in the life sciences. Cluster analysis is an important step in gene expression data analysis. It involves many statistical strategies and biological data concepts. The goal of this thesis is to build a data storage structure to store the cluster analysis related data. Our approach to this problem can be summarized as follows:

First, we studied the workflow of microarray technology and reviewed the cluster analysis concept. Based on the workflow, we identified a number of data requirements for our cluster analysis data storage.

Second, upon realizing the complexity of microarray data standard MAGE-OM, we designed our own conceptual data model for cluster analysis of gene expression data. In addition, a relational database schema was designed based on the object model.

Third, in order to integrate our design into a real microarray system, we reviewed public microarray databases and systems. Among three open source microarray systems, we chose a web-based microarray system – BASE, and designed a BASE-cluster relational database schema with a MySQL implementation to store the cluster analysis related data.

Finally, to validate our database and to populate the cluster structure and cluster results into the database, we reviewed three open source cluster analysis applications, and selected a Java-based clustering program MeV to implement. By loading a BASE-exported gene expression file, we are able to store the cluster structure and cluster results into the BASE-cluster database while hierarchical clustering is performed by MeV.

There are several advantages for storing the results of a cluster analysis to database compares to a text file. First, cluster results from different clustering methods can be easily compared and the results from a set of cluster operations can be also stored. Second, one cluster result can have different views by loading the data to different visualization tools, without performing a clustering algorithm again. Third, clusters of interests can be linked to the outside database or annotation tools.

6.2 Future work

Data analysis on gene expression data could become an exhausting work, since there are too many ways and too many tools for us to select. Of course there is no single "best" way for analysis. The proper and efficient data storage could allow investigators to query and mine the data by using different tools. Our cluster analysis database is built for such reasons. However, new statistical methods are emerging everyday. The data storage requirements for future cluster analysis are beyond our perspective. Although MeV has already included 14 clustering modules, we only implemented the hierarchical clustering module. Therefore, future work should include the following:

1. Refine our database by adding more data attributes required for the new statistical methods.
2. Implement all the clustering modules in MeV to populate cluster data to the database.
3. Build a bridge to connect BASE and MeV, either by plugging MeV into BASE, or invoking MeV locally through BASE and populating the desired data back to database server.
4. Build the interface to query the stored cluster data for future data comparison, visualization and annotation.

7 Reference

1. An adopted specification of the Object Management Group, Inc, Gene expression specification, version 1.0, formal/03-02-03, February 2003, http://www.omg.org/technology/documents/formal/gene_expression.htm
2. Anderle P, Duval M, Draghici S, Kuklin A, Littlejohn TG, Medrano JF, Vilanova D, and Roberts MA, Gene Expression Databases and Data Mining, *Biotechniques* 34: S36-S44 (March 2003)
3. Barbara D, An Introduction to Cluster Analysis for Data Mining, http://www.ise.gmu.edu/~dbarbara/755/cssurvey.pdf
4. Bassett Jr DE, Eisen MB and Boguski MS, Gene expression informatics --- it's all in your mine, *Nature Genetics Supplement* Vol.21, pp.51-55, 1999
5. Brazma A, Hingamp P, Quackenbush J, Sherlock G, Spellman P, Stoeckert C, Aach J, Ansorge W, Ball CA, Causton HC, Gaasterland T, Glenisson P, Holstege FC, Kim IF, Markowitz V, Matese JC, Parkinson H, Robinson A, Sarkans U, Schulze-Kremer S, Stewart J, Taylor R, Vilo J, Vingron M. (2001). Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nat Genet.* 29, 365-71.
6. Buhler J, Anatomy of a Comparative Gene Expression Study, http://www.cs.wustl.edu/~jbuhler//research/array/
7. Cheng Y and Church GM, Biclustering of expression data, *Proc Int Conf Intell Syst Mol Biol.* 2000; 8: 93-103
8. Churchill GA, Fundamentals of experimental design for cDNA microarrrays, Nature Genetics, V. 32 supplement pp 490-495, 2002

9. D'haeseleer P, Wen X, Fuhrman S, Somogyi R. Mining the Gene Expression Matrix: Inferring Gene Relatinships From Large Scale Gene Expression Data. *Information Processing in Cells and Tissues*, pp. 203-212, 1998

10. Dudoit S, Gentleman RC, and Quackenbush J, Open Source Software for the Analysis of Microarray Data, *BioTechniques* 34:S45-S51 (March 2003)

11. Eisen MB, Cluster and TreeView Manual, Stanford University, 1998-99

12. Eisen MB and Brown PO, DNA arrays for analysis of gene expression. *Meth. Enzymol.* **303**, 179–205 (1999).

13. Eisen MB, Spellman PT, Brown PO and Botstein D. (1998). Cluster Analysis and Display of Genome-Wide Expression Patterns. *Proc Natl Acad Sci U S A* 95, 14863-8.

14. Gardiner-Garden M and Littlejohn TG, A comparison of microarray database, *Briefings in Bioinformatics*, May 2001; 2, 2; Science Module

15. Han J and Kamber M. Data Mining: Concepts and Techniques, The Morgan Kaufmann Series in Data Management System, Jim Gray, Series Editor Morgan Kaufmann Publishers, August 2000

16. Hosack DA, Dennis Jr G, Sherman BT, Lane HC and Lempicki RA, Identifying biological themes within lists of genes with EASE. Genome Biology 2003, 4(9): R60

17. Kaminski N and Friedman N, Practical Approaches to Analyzing Results of Microarray Experiments, *Am. J. Respir. Cell mol. Biol.* Vol.27, pp.125-132, 2002

18. Kohonen T, *Self Organizing Maps* (Springer, Berlin, 1995).

19. Lazzeroni L and Owen A, Plaid Models for Gene Expression Data, Technical Report, Stanford University, March 2000.

20. Leung YF, Cavalieri D. Fundamentals of cDNA microarray data analysis. *Trends Genet.* 2003 Nov; 19(11):649-59

21. Nucleic Acid Hybridization, Graphics Gallery, About Biotech, Access excellence @ the national health museum, http://www.accessexcellence.org/AB/GG/nucleic.html

22. Pollard K, Lowe T& Dudoit S, Statistical Methods and Software for the Analysis of DNA Microarray Data, Bioconductor Tutorial, Aug.15, 2003, http://www.bioconductor.org/workshops/UCSC03/UCSCtalk.pdf

23. Raychaudhuri S, Stuart JM, & Altman RB, Principal components analysis to summarize microarray experiments: application to sporulation time series. *Pac. Symp. Biocomput.* **2000**, 455–466 (2000).

24. Saal LH, Troein C, Vallon-Christersson J, BASE v1.2x User Guide, Lund University, June 2003

25. Saal LH, Troein C, Vallon-Christersson J, Gruvberger S, Borg Å and Peterson C, BioArray Software Environment: A Platform for Comprehensive Management and Analysis of Microarray Data, *Genome Biology* 2002 3(8): software0003.1-0003.6

26. Saeed AI, Sharov V, White J, Li J, Liang W, Bhagabati N, Braisted J, Klapa M, Currier T, Thiagarajan M, Sturn A, Snuffin M, Rezantsev A, Popov D, Ryltsov A, Kostukovich E, Borisovsky I, Liu Z, Vinsavich A, Trush V, Quackenbush J.

TM4: a free, open-source system for microarray data management and analysis. *Biotechniques*. 2003 Feb; 34(2):374-8.

27. Sasik R, Hwa T, Iranfar N, and Loomis W.F., Percolation Clustering: A Novel Algorithm Applied to the Clustering of Gene Expression Patterns in Dictyostelium Development, *Pacific Symposium on Biocomputing* 6:335-347 (2001).
28. Scherf U, Ross DT, Waltham M, Smith LH, Lee JK, Tanabe L, Kohn KW, Reinhold WC, Myers TG, Andrews DT, Scudiero DA, Eisen MB, Sausville EA, Pommier Y, Botstein D, Brown PO, & Weinstein JN, A gene expression database for the molecular pharmacology of cancer, *Nature Genetics*, Vol.24, pp.236-244, 2000
29. Shannon W, Culverhouse R, and Duncan J, Analyzing microarray data using cluster analysis, *Pharmacogenomics* 4(1), 41-51, 2003
30. Spellman PT, Miller M, Stewart J, Troup C, Sarkans U, Chervitz S, Bernhart D, Sherlock G, Ball C, Lepage M, Swiatek M, Marks WL, Goncalves J, Markel S, Iordan D, Shojatalab M, Pizarro A, White J, Hubley R, Deutsch E, Senger M, Aronow BJ, Robinson A, Bassett D, Stoeckert Jr CJ, Brazma A. (2002). Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biology* 2003, 3(9): research0046.1-0046.9
31. Stoeckert Jr CJ, Causton HC and Ball CA, Microarray databases: standards and ontologies, *Nature genetics supplement* Vol.32 Dec. 2002, pp 469-473
32. Tavazoie S, Hughes JD, Campbell MJ, Cho RJ, & Church GM, Systematic determination of genetic network architecture. *Nature Genet*. 22, 281–285 (1999).

33. TIGR MultiExperiment Viewer Manual, Version 2.2, July 2003

34. TIGR MultiExperiment Viewer Algorithm API Description, DATANAUT Incorporated, Version 1.0, February 2002

35. The Central Dogma of Molecular Biology, Graphics Gallery, About Biotech, Access excellence @ the national health museum, http://www.accessexcellence.org/AB/GG/central.html

36. Tuimala J, M.Laine M, DNA Microarray Data Analysis, *CSC-Scientific Computing Ltd.* 2003

37. Yang YH and Speed T, Design issues for cDNA microarray experiments, *Nature Reviews Genetics* 3, pp.579 –588, 2002