9-1-2001

A New Approach to Clustering Biological Data Using Message Passing.

Huimin Geng

A New Approach to Clustering Biological Data Using Message Passing

A Thesis

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

University of Nebraska at Omaha

by

Huimin Geng

September, 2001

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

THESIS ACCEPTANCE

Acceptance for the faculty of the Graduate College, University of Nebraska, in partial fulfillment of the requirements for the degree of Master of Science, University of Nebraska at Omaha.

Committee

Name	Signature
Hesham Ali	
Bruce Chase	
Zhengxin Chen	

Chairperson (signature) _____ Date 9/20/04

Co-Chairperson (signature) _____ Date _____

(if applicable)

A New Approach to Clustering Biological Data Using Message Passing

Huimin Geng, MS

University of Nebraska, 2004

Advisor: Hesham Ali

ABSTRACT

Motivation: Clustering algorithms are widely used in bioinformatics, having been applied to a range of problems from the analysis of gene expression to the building of phylogenetic trees. Biological data often describe parallel and spontaneous processes such as molecular interactions and genome evolution. To capture these features, we propose a new clustering algorithm that employs the concept of message passing.

Methods: Inspired by a real-world situation in which people who have never met can form groups by exchanging messages, Message Passing Clustering (MPC) allows data objects to communicate with each other and produces clusters in parallel, thereby making the clustering process intrinsic. Other advantages of MPC over traditional clustering methods include that it is relatively straightforward to understand and implement and that it takes into account both local and global structure. We have proved that MPC shares similarity with Hierarchical Clustering (HC) but offers significantly improved performance.

Results: To validate the MPC method, we analyzed 35 sets of simulated dynamic gene expression data, achieving a 95% hit rate with 639 of 674 genes correctly clustered. We also applied MPC to real data sets to build a phylogenetic tree for 34 strains from nine species of *Mycobacterium* and to cluster 698 genes from a yeast cell-cycle database. The results show higher classification accuracies as compared to traditional clustering methods.

Supplementary information: http://bioinformatics.ist.unomaha.edu/~hgeng/

ACKNOWLEDGEMENTS

First and foremost, I wish to express sincere appreciation to my advisor, Prof. Hesham Ali, for his guidance, patience, and encouragement. What I have learned from him is not limited to research and academic standards but also about the importance of attitude to success in this world.

I would like to thank my committee members: Prof. Zhengxin Chen and Prof. Bruce Chase for their valuable comments and suggestions.

I would also like to acknowledge BIIG group members who have been a patient audience for a long time and have given me many suggestions: Dr. Bastola, Dr. Pauley, Xutao, Xiaolu, Alexander, and Dan.

Finally, this thesis would not be possible without the love and support from my family: Xutao, Zachary, and my parents.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

GLOSSARY

HC: Hierarchical Clustering.

MPC: Message Passing Clustering.

MS: Multidimensional Scaling.

NJ: Neighbor Joining.

ORFs: Open Reading Frames.

PCA: Principal Components Analysis.

SOM: Self-Organizing Map.

SVD: Singular Value Decomposition.

SVM: Support Vector Machines.

Chapter 1

INTRODUCTION

Clustering algorithms are frequently and successfully used in bioinformatics to organize multivariate data with similar patterns into distinct groups [Holter *et al.,* 2000], having been applied to problems ranging from the analysis of gene expression to the building of phylogenetic trees. A vast library of clustering techniques is available for different data attributes, goals, and personal preferences.

Among all available clustering methods, Hierarchical Clustering (HC) [Eisen *et al.,* 1998] is perhaps among those most widely accepted by biologists. In HC procedures, all data instances start in their own clusters, and the two clusters most closely related by some similarity metric are merged. This merging process is repeated until only a single cluster remains. HC is conceptually simple but often outperforms those algorithms which employ complex data structures and data flows. HC arranges the data into a tree structure which can be easily viewed and understood; and the hierarchical structure provides potentially useful information about the relationships among clusters.

However, HC has some inherent problems in dealing with the data generated from biological processes. HC always puts a priority on *global* distance without honoring *local* structures. Furthermore, in each clustering process, only the two clusters which have the globally highest similarity (or lowest distance) are merged, and clusters cannot be generated in parallel. But biological processes are often *parallel* by nature. For example, all organisms in an evolution system evolve in parallel to adapt to their local

environments, and multiple evolutionary events from various organisms may happen at the same time. In another example of a parallel process, behavior of multiple genes in a regulatory network can be monitored by microarray technology [Slonim, 2002]. At a specific time, each gene interacts only with its local environment. We believe that HC is not sufficient in reconstructing the phylogenetic tree or in clustering genes in a regulatory system, because the parallel process and local interaction will not be represented by HC, which is designed to sequentially merge objects by global measurements.

To overcome these problems, we propose a new clustering algorithm which simulates spontaneous and parallel biological processes. Inspired by a real-world situation in which people unknown to one another can form groups by exchanging information, MPC, by employing the concept of message passing in the operating system, allows data objects to communicate with each other, generates clusters in parallel so that the global attributes and local attributes are well-balanced, and improves the performance of clustering.

1.1. WHAT THIS THESIS IS ABOUT

This thesis is organized as follows. First we introduce the conception of microarray technology and microarray data analysis methods in Chapter 2. In Chapter 3 we discuss clustering analysis, including clustering problem formulation, input data formats, similarity metrics, similarity criteria and the assessment of clustering solutions. The computing model of MPC is illustrated by a real-world example in section 4.1. The algorithm of MPC is demonstrated in detail in 4.2. The properties of MPC and the proofs

are given in 4.3. The software package of MPC is introduced in 4.4. Chapter 5 shows the results of three case studies. The first data sets are simulated gene expression data; the second data set presenting the relatedness of DNA sequences is used to build the phylogenetic tree of 34 strains of nine *Micobacterium* species; and the last set is the gene expression data of 698 genes, over 72 conditions, from the Stanford yeast cell-cycle database. Chapter 6 summarizes the MPC method and the results of three case studies and provides a discussion.

1.2. THE CONTRIBUTIONS OF OUR METHOD

Our main contributions include proposing a novel clustering algorithm which employs the concept of message passing in the operating system, implementing this algorithm into a software package which can be freely downloaded from our web site, and giving possibilities for further development.

The MPC method is inspired by a real-word clustering situation, simulating human intelligence in the design. As a result, it often produces more accurate and meaningful clustering solutions than other popular clustering methods, such as the HC and NJ methods. The MPC method is especially good at describing parallel and spontaneous biological processes.

The MPC method is easy to understand, and its implementation is straightforward to implement. MPC shares similarity with the HC method, but it offers significantly improved performance because it takes into account both local and global structure. We

predict MPC should outperform most existing clustering algorithms when applied to biological data sets. The results in three case studies confirmed our hypotheses.

MPC, as a general purpose clustering algorithm, can be applied not only to biological data, but also to a wide range of fields such as city planning and WWW (World Wide Web) classification. In MPC, not every number K of clusters may be reached because multiple clusters are generated in parallel. This feature can provide us some information about the data and hence suggest the true number of clusters. Another advantage of MPC is its flexible structure which allows future development, e.g. changing the message type and the way the message is handled to deal with more challenging situations.

Chapter 2

BACKGROUND

This chapter a) covers concepts used throughout this thesis, including the importance of the microarray, how to design a microarray experiment, and the taxonomy of the microarray, and b) reviews microarray data analysis methods including detecting differential expression, pattern discovery, and class prediction and inferring regulatory networks.

2.1. CONCEPTS OF MICROARRAY TECHNOLOGY

DNA microarray technology has attracted tremendous interest among scientists, for it processes a very large number of genes (thousands of individual gene sequences with each sample spot less than 200 microns in diameter) while requiring only a dime-sized glass or a silicon chip. Using this new technology, researchers can get a better picture of the interactions among thousands of genes simultaneously in just a single experiment.

2.1.1. What are Microarrays

"A microarray is a tool for analyzing gene expression that consists of a small membrane or glass slide containing immobilized probes of many genes arranged in a fixed regular pattern", according to NCBI: http://www.ncbi.nlm.nig.gov.

The microarray is often used in, but not limited to, gene expression study. It can also be used at the genomic DNA or protein level. The sample or probe in a slide can be not only genes but also oligonucleotides, introns or intergenic regions or protein sequences depending on its use. The slides themselves are usually glass microscope slides, the size of two side-by-side pinky fingers, but can also be silicon chips or nylon membranes. The DNA is spotted or synthesized to the support in an orderly way so that researchers can identify a particular gene sequence by the location of each spot in the array.

2.1.2. Why Microarrays are Important

Ten years ago with traditional methods, molecular biologists had to work on "one gene in one experiment" basis. It was difficult to achive a high throughput of gene expression and to see the complete picture of gene function. Now, by using a new technology, the microarray, scientists can determine the expression levels of thousands of genes in a single experiment. With the help of a computer, the amount of mRNA bound to the spots on the microarray is precisely measured, generating a profile of gene expression in the cell. And gene-expression data can be analyzed by computational methods to extract meaningful information.

By taking advantage of the microarray, scientists will be able to ask increasingly complex questions and perform more intricate experiments. It's much easier to infer probable functions of new genes based on expression-patterns similarities with known genes, to reveal new patterns of coordinated gene expression across gene families, to

uncover entirely new categories of genes, and to understand how genes are co-regulated and the interrelationships of these genes. Furthermore, the use of microarrays, enabling scientists to examine a much larger number of genes, may speed the identification of genes involved in the development of various diseases. This technology will also aid the examination of the integration of gene expression and function at the cellular level, revealing how multiple gene products work together to produce physical and chemical responses to both static and changing cellular needs.

2.1.3. Designing a Microarray Experiment

The entire microarray process is based on hybridization probing, a technique that uses fluorescently labeled nucleic acid molecules as "mobile probes" to identify complementary molecules.

Let's consider two cells: a healthy cell and a diseased cell, both of which contain an identical set of genes, Gene-*1*, Gene-*2*, ..., and Gene-*n*. We are interested in determining the expression profile of this set of *n* genes in the two cell types. To do this, we isolate mRNA from each cell type and use this mRNA as templates to generate cDNA with a fluorescent tag attached. Red and green tags are used so that the samples from the healthy cell and the diseased cell can be differentiated. The two labeled samples are then mixed and incubated with a microarray containing the immobilized genes: Gene-*1*, Gene-*2*, ..., and Gene-*n*. After hybridization, the microarray is placed in a scanner that consists of lasers, a special microscope, and a camera. The fluorescent tags are excited by the laser, and the microscope and camera work together to create a digital image of the array.

These data are then stored in a computer and a special program is used to calculate the red-to-green fluorescence ratio which is usually represented in a matrix and used for further analysis to reveal some knowledge about the underlying biological processes.

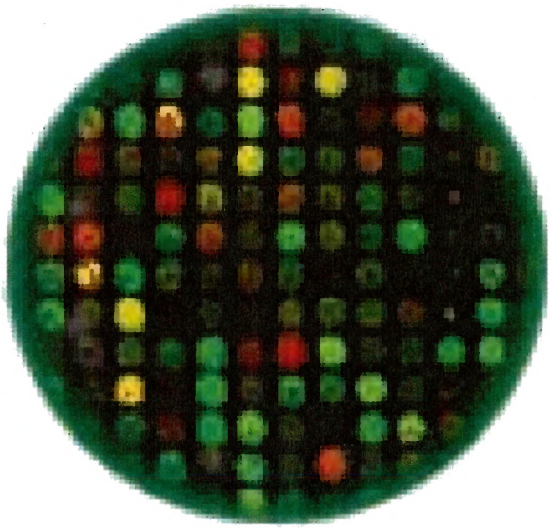


Figure 1. Gene expression image on the microarray.

- GREEN represents Control DNA derived from healthy tissue is hybridized to the target DNA.
- RED represents Sample DNA derived from diseased tissue is hybridized to the target DNA.
- YELLOW represents both Control DNA and Sample DNA are hybridized to the target DNA.
- BLACK represents neither Control DNA nor Sample DNA are hybridized to the target DNA.

From the above example, we can conclude the basic steps for designing a microarray experiment:

1. Prepare DNA chip using your chosen target DNAs.
2. Generate a hybridization solution containing a mixture of fluorescently labeled cDNAs.

3. Incubate hybridization mixture containing fluorescently labeled cDNAs with your DNA chip.
4. Detect bound cDNA using laser technology and store data in a computer.
5. Analyze data using computational methods.

In the literature there exist two confusing nomenclature systems for referring to hybridization partners. "Probe" and "target" are two common terms. A *probe* is a tethered nucleic acid with known sequence, whereas a *target* is a free nucleic acid sample whose identity/abundance is being detected.

2.1.4. Microarray Taxonomy

a) Experiment level:

According to the NCBI Gene Expression Omnibus, one microarray experiment can be modeled as three entities, called *platforms*, *samples* and *series*. A *platform* is a list of probes that define what set of molecules may be detected in any experiment. A *sample* describes the set of molecules that are being probed and references a single platform used to generate molecular abundance data. Each sample has one, and only one, parent platform that must be previously defined. A *series* organizes samples into the meaningful data sets that make up an experiment and are bound together by a common attribute.

Table 1. Microarray Taxonomy (**http://www.ncbi.nlm.nig.gov**):

Entity Type	Subtype	Description
Platform	Microarray	Spotted cDNA microarray
	Had	High density oligonucleotide array, e.g. Affmetrix Gene Chip™
	Filter	Filter or membrane array
	Aba	Antibody array
	Sage	Serial analysis gene expression (automatically generated) array
Sample	Dual channel	Dual mRNA target sample hybridization
	Single channel	Single mRNA target sample hybridization
	Dual channel	Dual DNA target sample hybridization, e.g. Microarray Comparative Genomic Hybridization (CGH)
	Sage	Serial analysis of gene expression
Series	Time-course	Time course experiment, e.g. yeast cell cycle
	Dose-response	Dose-response experiment, e.g. response to drug dosage
	Other ordered	Ordered, but unspecified
	other	Unordered

b) Application Level:

There are three basic types of samples that can be used in microarray experiments, two are genomic and the other is "transcriptomic" (measures mRNA level). They differ in the kind of immobilized DNA used to generate the array, and ultimately, the kind of information that is derived from the chip.

i. Changes in Gene Expression Levels (expression chip)

This kind of array is used to determine the level at which a certain gene is expressed. The immobilized DNA is cDNA derived from the mRNA of known genes, and the control and sample DNA hybridized to the chip is cDNA derived from the mRNA of normal and diseased tissue, respectively. Researchers can use expression chips to detect expression patterns (whether or not a particular gene is being expressed more or

less under certain circumstances) and to examine changes in gene expression over a series of time, e.g. within a cell cycle.

ii. Genomic Gains and Losses

DNA repair genes are considered to play a major role in cancer. Mutations within these genes often manifest themselves as lost or broken chromosomes, and certain chromosomal gains and losses are related to cancer progression. Measuring gains and losses in the copy number of chromosomal regions in tumor cells and analyzing obtained data, researchers can predict which chromosomal regions are most likely to harbor important genes for tumor initiation and disease progression.

Microarray Comparative Genomic Hybridization (CGH) is used to detect the gains and losses in the number of copies of a particular gene involved in a disease. In this technique, target DNA are large pieces of genomic DNA, and each spot of target DNA has a known chromosomal location. The control and sample DNA are fluorescently labeled genomic DNA from normal and diseased tissue respectively. If the number of copies of a particular target gene has increased, a large number of sample DAN will hybridized to those spots on the microarray that represent the gene involved in that disease. Therefore, those spots containing the disease gene will fluoresce red indicating that the number of copies of the gene involved in the disease has gone up.

iii. Mutations in DNA

To detect mutations or polymorphisms in a gene sequence, researchers can place the target sequence, which usually differ by only one or a few specific nucleotides on any

given spot within the microarray. This kind of experiment only requires only genomic DNA derived from a normal sample in the hybridization mixture.

Once researchers have established that a Single Nucleotide Polymorphism (SNP) pattern is associated with a particular disease, they can use SNP technology to determine whether an individual is susceptible to that disease. When genomic DNA from an individual is hybridized to an array loaded with various SNPs, the sample DNA will hybridize with greater frequency only to specific SNPs associated with that person. Those spots on the microarray will fluoresce with greater intensity, demonstrating that that person is susceptible to that disease.

2.2. A Review of Microarray Data Analysis Methods

However, the huge data sets generated from microarray experiments present us great challenge in interpreting and analyzing the data. It has become increasingly clear that simply generating the data is not enough; one must be able to extract from it meaningful information about the system being studied. A wide range of data-analysis approaches is available and depends on both the data and the goals of the experiment. There is no one-size-fits-all solution for the analysis and interpretation of genome-wide expression data. An understanding of both the biology and the computational methods is essential for tackling the associated "data mining" tasks, and the combined efforts of biologists, computer scientists, statisticians and software engineers make available a wealth of methods and tools for microarray data analysis.

In this section we summarize some of the common themes in microarray data analysis, including detection of differential expression, pattern discovery (also called *unsupervised* method), class prediction (also called *supervised* method), and the Bayesian network.

2.2.1. Detecting Differential Expression

The most basic question one can ask in a transcriptional profiling experiment is which genes' expression levels changed significantly. Among many different methods used for identifying significant changes, measuring differential expression by the ratio of expression levels between two samples is one of the simplest and the most widely used in transcriptional profiling experiments. Genes with ratios above a fixed cut-off k (that is, those whose expression underwent a k-fold change) were said to be differentially expressed [Eisen *et al.* 1998, MacQueen 1965, Ball and Hall 1967, and Herwig *et al.* 1999]. Now researchers rely on variants of common statistical tests which involve two parts: calculating a test statistic and determining the significance of the observed statistic [Everitt 1993, Out and Sayood 2003, and Saitou and Nei 1987].

2.2.2. Pattern Discovery

Pattern discovery provides a high-level overview of a data set and may be the first analysis step in a study that ultimately involves other analytical methods. Such techniques include a variety of dimension-reduction methods, as well as various 'clustering' techniques. These methods simplify the data set, and they are considered

unsupervised, meaning that the reduction is derived solely from the data rather than reflecting any previous knowledge or classification scheme.

2.2.2.1. Dimension-Reduction Methods

Principal Components Analysis (PCA) [Raychaudhuri *et al.* 2000, and Landgrebe *et al.,* 2002], Singular Value Decomposition (SVD) [Alter *et al.* 2002, and Holter *et al.,* 2000] and Multidimensional Scaling (MS) [Bittner *et al.* 2000, and Khan *et al.* 1998] are related dimension-reduction techniques that can be used for visualizing large data sets. Each of these approaches projects the data into a new space based on linear combinations of variables that retain a large amount of the original data's variation. These techniques rely on the idea that most of the data's variation can be explained by a smaller number of transformed variables. However, data-reduction and visualization tools are projecting thousands of dimensions into two or three, so much information may be lost and the reduced data may fail to capture the expected aspects of a data set.

2.2.2.2. Clustering Methods

The term 'clustering' applies to a wide variety of unsupervised methods for organizing multivariate data into groups with roughly similar patterns. Clustering is a fundamental problem which has many applications in expression-data analysis and elsewhere in biology, e.g. in the building of phylogenetic trees. Clues to unknown gene function may be inferred from clusters of genes similarly expressed across many samples.

Cluster analysis may be used primarily for data reduction and visualization, or it may be used to generalize or predict the categorization of new samples.

a) Hierarchical Clustering Methods

Perhaps most familiar to biologists are the hierarchical clustering methods [Eisen *et al.* 1998, and Spellman *et al.* 1998]. In this family of techniques, all data instances start in their own clusters, and the two clusters most closely related by some similarity metric are merged. The process of merging two clusters is repeated until a single cluster remains. This arranges the data into a tree structure that can be broken into the desired number of clusters by cutting across the tree at a particular height. Tree structures are easily viewed and understood, and the hierarchical structure provides potentially useful information about the relationships between clusters. However, as later merges often depend on aggregated measures of clusters containing many scattered elements, interpreting the broadest clusters can be difficult.

b) K-mean and SOM Clustering Methods

Another common family of clustering methods is partition (or centroid) algorithms. These methods generally require specification of the numberof clusters, k, and start with k data points that may be chosen either randomly or deliberately. The algorithm then partitions the samples into the k clusters, optimizing some objective function (such as within-cluster similarity) by iteratively assigning samples to the nearest centroid's cluster and adjusting the centroids to represent the new clusters' center points.

The self-organizing map (SOM) [Kohonen 1997, Tamayo *et al.* 1999] is a variation that allows samples to influence the location of neighboring clusters. Such maps are particularly valuable for describing the relationships between clusters. New centroid methods specifically for microarray data have also been proposed [Ben-Dor *et al.* 1999]. Generally the number of "true" clusters in the data is not known. Therefore, it is a good idea to run the partition algorithms with different values for k that are near the number of clusters one expects from the data to see how the sum of distances reduces with increasing values of k.

2.2.2.3. Other Techniques

Other techniques abound. Some seek to optimize a measure of within-cluster similarity or between-cluster separation, but they avoid specifying the number of clusters ahead of time, instead specifying information-theoretic bounds on cluster membership [De Smet *et al.* 2002, Heyer *et al.* 1999, and Sharan and Shamir 2000]. Model-based methods assume the data can be generated by a specified statistical model and search for model parameters that best fit the data [Fraley and Raftery 2002, Yeung *et al.* 2001]. So-called *fuzzy* clustering finds groups, but may allow elements to appear in more than one cluster or in no clusters at all [Hastie *et al.* 2000].

Choosing a clustering method is still very much dependent on the data, the goals, and the investigator's personal preferences. One caveat, however, is to be sure that the primary goal is the pattern discovery or dimension reduction that clustering offers. If the

intent is to distinguish between currently defined sample classes (e.g., tumors and normal tissue samples), the class prediction methods described below may be more effective.

2.2.3. Class Prediction

In contrast to pattern discovery, class-prediction methods are techniques specifically designed to classify objects into known groups. A wealth of machine-learning literature describes computational techniques for classifying multidimensional data. Most such methods include a training-phase run on samples whose classes are already known, and a testing phase, in which the algorithm generalizes from the training data to predict classifications of previously unseen samples. Because of this directed training phase, prediction methods are referred to as *supervised* classification methods.

Gene-expression data presents unusual challenges for machine-learning algorithms, which are generally designed to work with large numbers of samples over relatively few variables. In contrast, a typical microarray experiment measures thousands of genes (variables) but includes only tens or hundreds of patients' samples. Most algorithms stumble when faced with problems of these dimensions. A common problem is *overfitting* the data, where an algorithm models the training samples too exactly, without sufficient generalization. In consequence, classification of the training examples may well be perfect, but subsequent attempts to classify new test data fail dismally. A high level of noise in the microarray data is another problem. Sometimes training examples are misclassified. All of these traits make sample prediction from array data particularly challenging.

2.2.3.1. Support Vector Machines

Support vector machines (SVMs) are a family of statistical machine-learning methods that have been proposed as particularly suitable to the dimensions of microarray learning problems [Brown *et al.* 2000, Furey *et al.* 2000]. Intuitively, SVMs try to draw a hyperplane in *n*-dimensional gene-expression space between the training examples from two classes. If no separating hyperplane exists, the samples are mapped into a higher dimensional space where such a separator does exist. The algorithms minimize potential overfitting problems by choosing the separator farthest from the training samples, thus leaving room for generalization. More-complex mapping functions provide nonlinear mappings into higher dimensional spaces, resulting in a nonlinear classifier for the original data.

2.2.3.2. Decision Tree

Decision tree algorithms classify samples by filtering them through a tree-like structure, testing at each branchpoint some simple attribute of that sample. Tree models are easily built, easily understood, and able to model quite-complex functions, but single decision trees are particularly prone to overfitting. Many modified tree-based techniques for avoiding overfitting and improving performance are available, such as *pruning* the tree, i.e. restricting the number of consecutive branches so that it is forced to generalize. *Bagging* and *boosting* techniques are more powerful solutions to avoid overfitting, by

repeatedly sampling the data to build many trees and combining these trees into a single predictive model.

2.2.3.3. Other Prediction Methods

Other prediction methods include linear and quadratic discriminant methods [Dudoit *et al.* 2000], weighted voting [Golub *et al.* 1999], shrunken centroids [Tibshirani 2002] and compound covariates [Hedenfalk *et a*l. 2001]. A deceptively simple but powerful approach is *k*-nearest-neighbor prediction, in which the prediction for a test sample x is the most common class label among the k training samples most similar to x.

In deciding how best to approach a prediction problem, it is first important to consider the desired outcome. K-nearest neighbors, weighted voting, and SVMs were all comparable, but a combination of several methods outperformed any single predictor. Overfitting may be avoided by using simpler SVM mapping functions when complex ones are not needed, or by limiting the number of iterations of boosting algorithms. There is some consensus that simpler methods outperform complex ones in the typical case where the number of genes is much larger than the number of samples.

Neural networks, decision trees and k-nearest neighbors can, in principle, separate any number of output classes. SVMs and various linear methods are inherently binary — they distinguish between two classes only. It is, however, possible to combine binary predictors in order to separate multiple classes.

2.2.4. Inferring Regulatory Pathways and Networks

More complex gene relationships may be discovered as we learn to combine the data in more complex ways. Bayesian network models and variations are now the focus for a growing number of researchers concerned with discovering novel interactions, information dependencies and regulatory relationships from expression data. More recent modifications of Bayesian network methods focus further on finding probabilistically supported gene interactions or on combining these into subnetworks [Hartemink *et al.* 2002, Pe'er *et al.* 2001], on modeling 'latent' or hidden variables representing biological information unavailable to the model [Segal *et al.* 2001, Yoo *et al.* 2002], and on incorporating prior biological knowledge or annotation [Hartemink *et al.* 2002, Hartemink *et al.* 2001]. Current pathway methods seem to do reasonably well in finding correlated sets of genes. However, it has proved more difficult to infer the direction of causal relationships successfully directly from transcriptional data. In general, models that incorporate existing constraints from other data sources seem to produce hypotheses that agree better with existing biological knowledge than do models learned from the expression data alone [Hartemink *et al.* 2002].

Chapter 3

CLUSTERING ANALYSIS

Most clustering algorithms are distance- or similarity-based. Clustering performance relies on similarity metrics and similarity criteria. Similarity metrics define how to calculate the distance between each pair of nodes. Similarity criteria define how to calculate the distance of each pair of clusters. At the end of this chapter, we introduce how to evaluate the clustering solutions.

3.1. PROBLEM FORMULATION:

Let $O = \{O_1, O_2 \ldots O_n\}$ be a set of n objects and let $C = \{C_1, C_2 \ldots C_k\}$ be a partition of O into k subsets such that $Ci \cap Cj = \phi$ and $\bigcup_{i=1}^{k} Ci = O$. Each subset C_i is called a cluster, and the partition C is a clustering solution. Given a raw data matrix or similarity (or dissimilarity) data matrix, the goal of clustering is to find a partition C of a set of objects O such that the resulting clusters are homogeneous and well-separated. This means that objects assigned to the same cluster are as similar to each other as possible, while objects belonging to different clusters are as dissimilar with each other as possible.

Note: here we use hard clustering which means that each object is assigned to one and only one cluster. In fuzzy clustering, each object can be assigned to more than one cluster.

3.2. INPUT DATA FORMATS:

There are two kinds of input data formats for a given clustering problem: fingerprint data and similarity data.

Fingerprint data (also called profile data) is a data matrix with each object associated with a real-valued vector, called its fingerprint or pattern, which contains m measurements on the object. For example, if there are n genes represented by the rows and m experiments represented by the columns, then the entry (i, e) in the data matrix D represents the expression level of an mRNA of gene i at condition e, where $1 \leq i \leq n$ and $1 \leq e \leq m$. The i^{th} row in the data matrix D represents the expression vector of gene i over all m experiments. When clustering genes, the objects needed to be clustered are genes.

Similarity data (or dissimilarity data) represents pairwise similarity (or dissimilarity) values between objects. For example, if there are n genes, the entry (i, j) in the similarity (or dissimilarity) matrix S represents the similarity (or dissimilarity) of gene i and gene j, where $1 \leq i \leq n$ and $1 \leq j \leq n$. Similarity (or dissimilarity) data can be computed from fingerprint data, i.e. by correlation between vectors for similarity and by computing distances to get the dissimilarity. However, the fingerprint data cannot be fully recovered from the similarity (or dissimilarity) data. The fingerprint matrix is of order $n \times m$ (n genes and m conditions in the gene-expression profile), while the similarity matrix is of order $n \times n$ where n is the number of genes. Though fingerprint data contains more information, similarity (or dissimilarity) data is completely generic and can be used as the input to clustering in any application.

3.3. SIMILARITY METRICS

Many different similarity metrics can be used in clustering problems, among which the two most popular are Euclidean distance (a dissimilarity measure) and the correlation coefficient (a similarity measure).

Correlation coefficients range from -1 to 1. Two genes having correlation coefficient 1 are said to be *correlated* (0: *uncorrelated*; -1: *anti-correlated*). A high correlation coefficient implies high similar expression patterns between two genes. Euclidean distances are equal to or greater than 0. A high Euclidean distance indicates low similarity between two genes.

The following formulas are most commonly used to calculate the correlation coefficient [Eisen *et al.*, 1998] and Euclidean distance between two data objects:

$$S(X,Y) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{Xi - \overline{X}}{\Phi_X}\right)\left(\frac{Yi - \overline{Y}}{\Phi_Y}\right), \quad \Phi_X = \sqrt{\sum_{i=1}^{m}\frac{(Xi-\overline{X})^2}{m}}, \quad \overline{X} = \sum_{i=1}^{m} Xi/m \tag{1}$$

$$D(X,Y) = \sqrt{\sum_{i=1}^{m}(Xi-Yi)^2} \tag{2}$$

X and Y represent m dimensional data objects; X_i and Y_is represent the value of X and Y, on the i^{th} coordinate; $\overline{X}$ and $\overline{Y}$ are the mean of all Xi's and Yi's; Φ_X and Φ_Y are the standard deviation of Xi's and Yi's; $S(X,Y)$ and $D(X,Y)$ represent similarity and dissimilarity between genes X and Y respectively.

3.4. SIMILARITY CRITERIA

The dissimilarity (distance) between two clusters depends on the clustering criterion. There are four major distance criteria used in the HC method and in the MPC method: single linkage, complete linkage, average linkage, and centroid linkage [Everitt 1993].

In single (Figure 2), complete (Figure 4) and average linkage (Figure 4), the distance of two clusters is defined as the minimum distance, the maximum distance, and the average pairwise distance between all pairs of data objects (each pair has one data object from each of the two clusters); in centroid linkage (Figure 5), the distance between clusters is defined as the distance between the cluster centroids (mean vectors of the clusters).

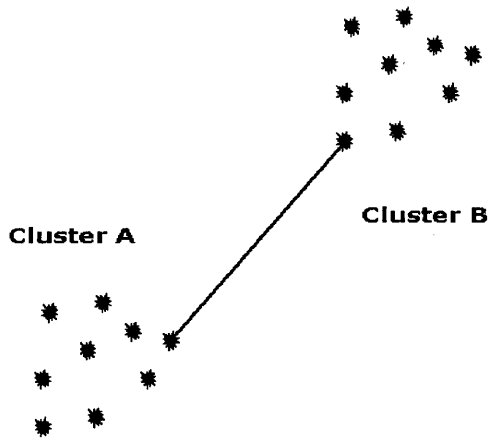


Figure 2. Single Linkage.

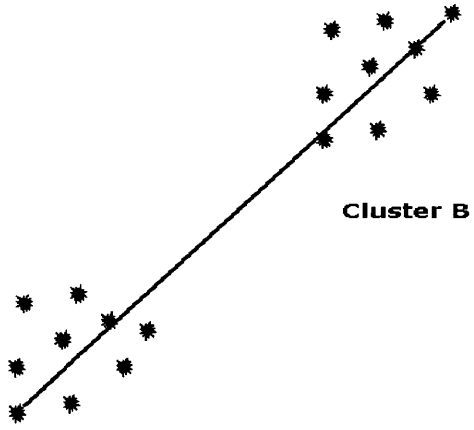


Figure 3. Complete Linkage.

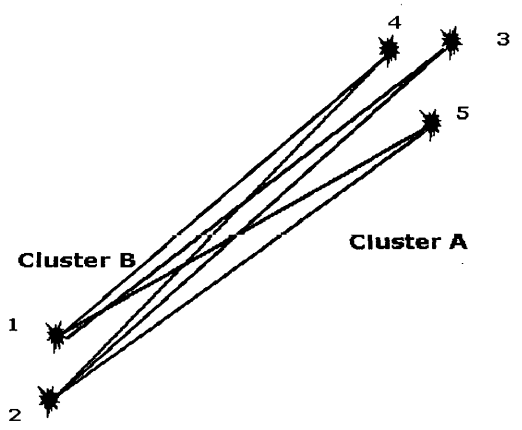


Figure 4. Average Linkage.

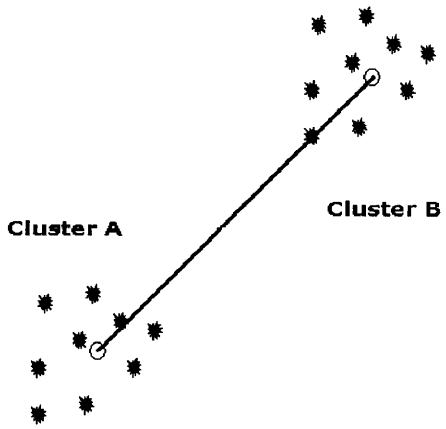


Figure 5. Centroid Linkage.

The dissimilarity (distance) between a newly merged cluster $C_i \cup C_j$ and any other cluster C_k is defined by [Lance and Williams 1967], where α_i, α_j, β, and γ are appropriately chosen parameters.

$$d(Ci \cup Cj, Ck) = \alpha_i d(Ci, Ck) + \alpha_j d(Cj, Ck) + \beta d(Ci, Cj) + \gamma \mid d(Ci, Ck) - d \mid (Cj, Ck) \quad (3)$$

The following table generalizes the values of the parameters for various clustering criteria, where n_i denotes the number of objects in cluster C_i.

Table 2. Dissimilarity Criteria and Parameters.

Criterion	α_i	β	γ
Single	1/2	0	-1/2
Complete	1/2	0	1/2
Average	$n_i/(n_i+n_j)$	0	0
Centroid	$n_i/(n_i+n_j)$	$-n_in_j/(ni+nj)^2$	0

Examining these criteria more closely we get:

Single: $$d(Ci,Cj) = \min\{d(s,t) \mid s \in Ci, t \in Cj\} \quad (4)$$

Complete: $$d(Ci,Cj) = \max\{d(s,t) \mid s \in Ci, t \in Cj\} \quad (5)$$

Average: $$d(Ci,Cj) = \frac{\sum_{s \in Ci} \sum_{t \in Cj} d(s,t)}{n_s * n_t} \quad (6)$$

Centroid: $$d(Ci \cup Cj, Ck) = \frac{n_i * d(Ci,Ck) + n_j * d(Cj,Ck)}{n_i + n_j} - \frac{n_i * n_j * d(Ci,Cj)}{(n_i + n_j)^2} \quad (7)$$

Table 3 summarizes additional properties of those inter-group distance measures [Everitt 1993].

Table 3. Standard Agglomerative Hierarchical Clustering Methods.

Method	Remarks
Single linkage Sneath (1957)	Tends to produce unbalanced and straggly clusters (chaining), especially in large data sets. Does not account for cluster structure.
Complete linkage Sorensen (1948)	Tends to find compact clusters with equal diameters (maximum distance between objects). Does not account for cluster structure.
Average linkage Sokal and Michener (1958)	Tends to join clusters with small variances. Intermediate between single and complete linkage. Accounts for cluster structure. Relatively robust.
Centroid linkage Sokal and Nichener (1958)	Assumes points can be represented in Euclidean space (for geometrical interpretation). The more numerous of two groups clustered dominates the merged cluster, subject to reversals.

3.5. ASSESSMENT OF CLUSTERING:

A key problem in the design and analysis of clustering methods is how to evaluate solutions. Intuitively we know that, a good clustering method should ensure that objects in the same cluster are similar to each other and different from objects in other clusters. Many cluster-evaluation metrics have been designed to formalize this intuition. There is, however, no single best way to evaluate a clustering method, and no single best clustering method for a data set. How best to compare clustering approaches depends on the purpose of clustering and on the information available. For example, there are different measures depending on whether a true solution is known.

3.5.1. True solution is given

If the true solution is given, then we wish to compare the suggested solution to the true solution. Any clustering solution can be represented by a binary $n \times n$ matrix C, in which $C_{ij} = 1$ if and only if i and j belong to the same cluster in that solution. Let T_{ij} and C_{ij} be the true solution and suggested solution. Let n_{kt} denote the number of pairs *(i, j) (i* $\neq j$*)* for which $T_{ij} = k$ and $C_{ij} = t$, where $k, t = 0$ or 1. So n_{11} denotes the number of true mates in the suggested solution which are also mates in the true solution; n_{00} denotes the number of nonmates in the suggested solution which are not mates in the true solution; n_{01} denotes the number of true mates in the suggested solution which are not mates in the true solution and n_{10} counts the nonmates in the suggested solution which are true mates in the true solution.

The Minkowski measure [Sokal, 1997] is defined as $\sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}}$. It measures the proportion of disagreements to the total number of mates in the true solution. A perfect solution has a score zero, and the lower the score, the better the solution.

The Jaccard coefficient [Everitt, 1993] is defined as $\frac{n_{10}}{n_{11} + n_{10} + n_{01}}$. It is the proportion of correctly identified mates to the sum of the correctly identified mates plus the total number of disagreements. A perfect solution has a score of one, and the higher the score, the better the solution.

Also, sensitivity is defined as $\frac{n_{11}}{n_{11} + n_{10}}$, and specificity is defined as $\frac{n_{11}}{n_{11} + n_{01}}$.

Here, n_{00} is not involved in the measures, since solution matrices tend to be sparse and this term would dominate the other three in good and had solution alike. When the true solution is known only for a subset $N^* \subset N$, the Minkowski and Jaccard measures can be computed on the sub-matrices corresponding to N*.

3.5.2. True solution is unknown

If we don't know the true solution, then we would like to evaluate the suggested solution by satisfying two criteria: homogeneity (an intra-cluster measure --- objects in the same cluster are highly similar to each other) and separation (an inter-cluster measure --- objects from different clusters have low similarity to each other). The definitions of average homogeneity and average separation are given as follows [Lance and Williams 1967]:

$$H_{Ave} = \frac{1}{|N|} \sum_{X \in N} S(X, Cl(X)), \tag{4}$$

$$S_{Ave} = \frac{1}{\sum_{i \neq j} |C_i||C_j|} \sum_{i \neq j} |C_i||C_j| S(C_i, C_j), \tag{5}$$

We abuse the notion of X representing a data object and its vector, and $Cl(X)$ representing the cluster of X and the centroid vector of the cluster. N is the set of all data objects in the cluster and $|N|$ means the number of data objects in the set N. C_i and C_j are disjoint clusters and their vectors in the solution. Recall that S is the similarity between two data objects. H_{Ave} measures the average similarity between the vector of an object

and that of its cluster. S_{Ave} measures the weighted average similarity between clusters' vectors. We say a clustering solution has high quality if H_{Ave} is relatively high and/or S_{Ave} is relatively low.

Also, we can measure the worst case instead of the average case using minimum homogeneity and maximum separation. $H_{min} = min\{S(F(u), F(Cl(u)))\}$, $S_{max} = max\{S(F(u), F(Cl(u)))\}$. In computing all the above measures singletons are considered as additional one-member clusters.

Note that since homogeneity and separation can be defined in different ways, this leads to different mathematical formulations. Moreover, even the definition of homogeneity and separation are precisely known, those two objectives often conflict with each other: the improvement in one will worsen the other.

3.5.3. General Comments on Evaluations

There are different measures in different situations and different techniques often highlight different patterns in the data, so complementary methods may be helpful in analyzing a single data set.

As we have seen, if the 'true' clustering is known (simulated data), a good metric might measure the fraction of co-clustered pairs from the true solution that are grouped together by the new method. If clustering is to be used primarily for data reduction, from that point of view, the best clustering is the one that allows expression of the entire data set in minimal space. This generally requires making assumptions about data distributions and the data representation format. If clusters are to be used to predict classifications of

other samples, one might choose to evaluate each clustering by its predictive power. Another desirable property of a clustering is stability; that is, if the experiment were repeated again, one would hope to obtain similar clusters. A standard technique for testing cluster reliability involves adding a small amount of noise to the data and reclustering.

Choosing the right number of clusters is crucial for many hierarchical and partitioning algorithms. One approach to setting the desired number of clusters is based on the observation that good clusters will probably not change dramatically if a small randomly chosen subset of the samples is discarded. Repeated sampling can be used to determine the number of clusters that provides the most stable solution. Another approach compares a measure of within-cluster distances from the cluster centroids to its expectation, and chooses the number of clusters maximizing the difference.

Chapter 4

MESSAGE PASSING CLUSTERING

In this chapter, we will illustrate the essences of the proposed method in detail. First, we demonstrate our computing model with an interesting real-world situation. Then we describe our algorithm: the inputs, the outputs, the main function and each subroutine of the program. Next, the properties of the MPC method are pointed out. Lemmas and theorems of MPC and the proofs are given so that it's easy to understand why MPC and HC share similarity, but more often than not, the two approaches generate different solutions in favor of the MPC method. At the end of this chapter, we introduce the software implementation of the MPC and the HC methods, which can be obtained from our home page.

4.1. COMPUTING MODEL OF MPC

The computing model of the MPC method is inspired by an interesting real-world situation. Suppose we have a party where people don't know one another in the beginning. Then each person may look around and talk to other persons to see if they share some common interest. If so, they may continue the conversation and other people with the same interest may also join this group later. It is often the case that after a while a set of clusters of talking groups is formed at the party.

Figure 6. Party Model.

This party model shows a parallel and spontaneous clustering process by exchanging information between people. We can see that many biological examples share great similarity with this party model. In the MPC algorithm, we employ the concept of message passing to represent the information exchanging processes between data objects.

4.2. ALGORITHM OF MPC

We will present the MPC algorithm in pseudocode style that is similar to C++ but ignores restrictions of C++ language.

Inputs: M-dimensional vectors $X_1, X_2, \ldots, X_n$

Outputs: A partition of $X_1, X_2, \ldots, X_n$ into K clusters

```
MPC (X1, X2,...,Xn)
{
    Initialize (X1, X2... Xn);
    while (Num_Clusters>K)
    {
        for (each cluster Ci) //message sending
        {
            Cj=Find_Nearest_Neighbor (Ci);
            Msg_Send (Ci,Cj);
        }
        for (each cluster Ci) //message receiving
        {
            Cj=Msg_Rcv (Ci) //check message
            if (Cj not equal to NULL)
            {
                New Cluster C'= Merge (Ci, Cj);
                Num_Clusters= Num_Clusters-1;
            }
        }
    }
}
Initilize (X1, X2,...,Xn)
{
    Num_Clusters=n; //Number of Clusters
    for (each cluster Ci)
    {
        Ci=Xi;      // Assign the vector to a cluster
```

```
            Ci.FROM=NULL;
            Ci.TO=NULL;
        }
}
Msg_Send(Ci, Cj) //send a message from Ci to Cj
{
        Ci.TO=j;
        Cj.FROM=i;
}
Msg_Rcv (Ci) //check the message box
{
        if ( Ci.FROM=Ci.TO=j) //find mutually
            return Cj;
        else
            return NULL;
}
```

Like all clustering algorithms, the input to the MPC program is a set of *M*-dimensional vectors. The size of the set is *N* and the output is a partition of the *N* vectors into *K* clusters where *K* can be any number from 1 to *N*. The key idea of MPC is to allow vectors to communicate with each other so that the clusters can be formed in parallel.

Initially each vector is a cluster in its own, so the number of clusters is *N*. Each cluster *Ci* is associated with two special memory cells, ***Ci.TO*** and ***Ci.FROM***. These two cells function as a message box with *Ci.TO* storing the outgoing message and *Cj.FROM* storing the incoming message. During the clustering process, each cluster will send a message to its nearest neighbor (the cluster with maximum similarity to the sending cluster) by calling function ***Msg_Send()***. On the other hand, each cluster may receive a

message from others by calling ***Msg_Rcv()***. Now the content of the massage is the ID (identification number) of the cluster. Later in the discussion section we will talk about sending a more complicated message attached to the ID as a tail. After a round of sending and receiving messages, each cluster checks the message box. If the outgoing and incoming addresses are the same, a mutually nearest neighbor (*Ci* is the nearest neighbor of *Cj* and *vice versa*) is found such that the two clusters merge to one. In MPC, each round of message exchanging allows each cluster to identify its nearest neighbor; if two clusters find they are a *couple* (mutually nearest neighbors), then they "marry" (merge) to form a new bigger cluster. This process is repeated until the specified number of clusters *K* is reached. The functions ***Find_Nearest_Neighbor()*** and ***Merge()*** are trivial, so no pseudocode for them are given here.

4.3. PROPERTIES

We show some properties of the MPC algorithm. While it may appear that MPC and HC produce similar outputs, we show that, more often than not, the output clusters of the two approaches differ in favor of the MPC method.

Definition: A *couple* is a pair of clusters *a* and *b* such that they are mutually nearest. A couple (*a,b*) should satisfy the following two conditions, where *i* is any cluster in the current cluster pool *P* and *D(a, b)* is the distance between the cluster *a* and b.

$$b = \underset{i \in P}{arg\,min}\, D(a,i)$$

$$a = \underset{i \in P}{arg\,min}\, D(b,i)$$

Theorem 1: The number of new clusters produced at each round of message exchanging is from 1 to n/2, where n is the number of clusters at the previous step.

Proof of theorem 1: The number of potential couples in the cluster pool is from 1 to n/2, so the proof follows immediately from the MPC algorithm. ■

The theorem shows clusters merge in an ideally exponential fashion so that MPC is a conceptually fast algorithm which reflects its parallel process.

Lemma: Let (a,b) be a couple in the current cluster pool P, and let the similarity criteria be single, complete, or average linkage; $D(a,b)$ is bounded by the following inequalities:

$$D(a,b) \le D(a,(b \cup c)) \quad \text{(i)}$$
$$D(a,b) \le D((a \cup b),c) \quad \text{(ii)}$$
$$D(a,b) \le D(a,(c \cup d)) \quad \text{(iii)}$$

where c and d can be any clusters except a and b in the current disjoint cluster pool P.

Proof of Lemma: Since *couple* is a symmetric definition, we can exchange a and b in the preceding lemma. Now, assuming the similarity criterion is complete linkage, we can easily prove this lemma.

Proof of (i):

$$D(a,(b \cup c)) = max(D(a,b),D(a,c)) \quad \text{(Def. of complete linkage)}$$
$$\ge D(a,b)$$

Proof of (ii):

$$D((a \cup b), c) = max(D(a,c), D(b,c)) \qquad \text{(Def. of complete linkage)}$$
$$\geq D(a,c)$$
$$\geq D(a,b) \qquad \text{(Def. of couple)}$$

Proof of (iii):

$$D(a,(c \cup d)) = max(D(a,c), D(a,d)) \qquad \text{(Def. of complete linkage)}$$
$$\geq D(a,c)$$
$$\geq D(a,b) \qquad \text{(Def. of couple)}$$

■

For single and average linkage, the lemma can be proved in similar fashion. It is important to note that the lemma does not apply to the case of centroid linkage because the centroid of a cluster is able to "move" during the clustering process by merging with other clusters, and hence the distance between two clusters is not "fixed". We can see this clearly in Figure 1.

Theorem 2: Comparing MPC and HC in various similarity criteria, the following propositions hold:

1) In the case of centroid linkage, the clustering dendrogram from MPC and HC are generally different.

2) In the case of single, complete, or average linkage, the final clustering dendrograms from MPC and HC are equivalent.

3) In the case of single, complete, or average linkage, intermediate clusters are generally different. If the number of final clusters is specified in advance, MPC and HC may yield different solutions.

Proof of 1): Show by example: Suppose there are four 2-dimensional data objects a, b, c, and d with coordinates given in Figure 7. MPC merges (a b) and (c d) at the same step, while HC merges only (a b) at the first step, then updates the distance table. Now the distance between c and the centroid of (ab) is less than the distance between c and d, so c was grouped with (a b) but not with d at the second step. Different clusters generated by MPC and HC are shown in Figure 8. ■

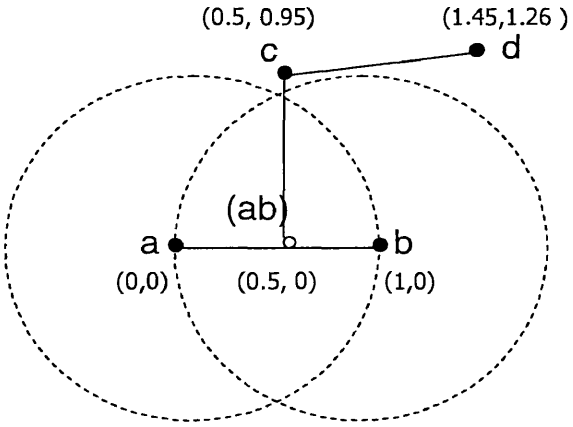


Figure 7. Coordinate of Nodes.

Table 4. Distance Tables: (i) first step; (ii) second step.

	b	c	d
a	1	1.07	1.92
b		1.07	1.34
C			1.01

(i)

	c	d
(ab)	0.95	1.58
c		1.01

(ii)

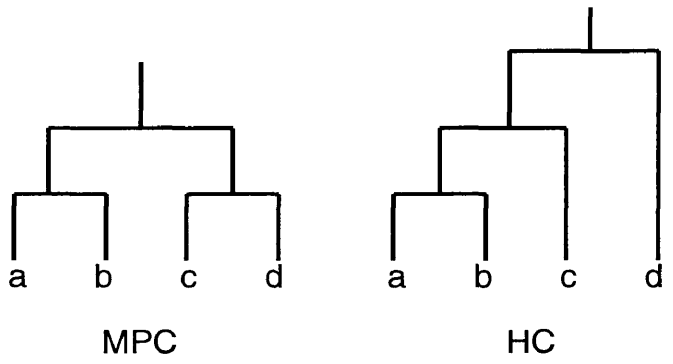


Figure 8. Dendrograms from MPC and HC.

We say the solution given by MPC is superior to that given by HC in this case, because it conforms to our intuition.

Proof of 2): A dendrogram is a tree which represents the merging processes controlled by the clustering algorithm. A branch in the dendrogram represents a marriage of two data objects in a couple. Now we show by induction that MPC and HC produce the same dendrogram.

Basis: In the leaves level, each data object represents a cluster. MPC and HC have the same dendrogram which are just a set of data objects.

Induction: Assume that at some step, MPC and HC have exactly the same dendrogram. We want to show that a branch produced by merging a couple in MPC should have a correspondence in HC. Suppose we identify a couple *(a, b)* which is about to merge in MPC. From lemma (c) $D(a,b) \leq D(a,(c \cup d))$. We see that *(a, b)* remains a couple even if any other couple *(c, d)* may be merged before the marriage of *(a, b)* in HC. So the same branch produced by the merging of *(a, b)* will be preserved in HC. ■

Proof of 3): Show by example: Figure 9 gives us an example showing different solutions from MPC and HC when the number of final clusters is three. In HC, there are two singletons, and we know singletons are bad in clustering problems. While in MPC, d and e are grouped if we consider not only global distance but also local area. The distance between d and e is large, but d and e are, relatively, closest to each other as compared to other nodes. ■

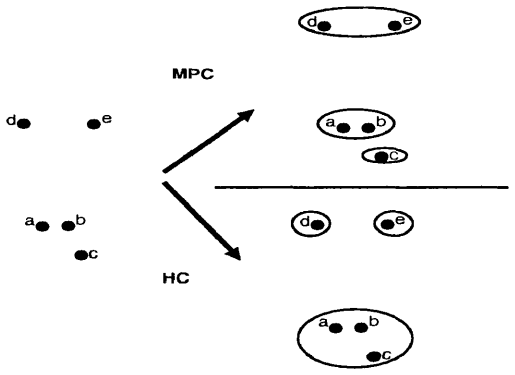


Figure 9. Clustering Results from MPC and HC.

4.4. SOFTWARE PACKAGE

Software implementation of the proposed algorithm and the hierarchical algorithm can be obtained at *http://bioinformatics.ist.unomaha.edu/~hgeng/*.

The software is written in C++. The program input is a matrix of the fingerprint data. Users are allowed to select:

- parameters of clustering algorithms: MPC or HC;

- similarity metrics: correlation coefficient or Euclidean distance;
- similarity criteria: single, complete, average, or centroids;
- and the cluster display: with or without branch length.

Two files are outputted:

- the 'tree' file is used as an input to a tree view software, NJplot [Perrière and Gouy 1996], to display the clustering dendrogram with/without branch lengths;
- the 'cluster' file contains the information of clustering processes, including the number of clusters, the size of each cluster, the data objects in each cluster, and the average homogeneity and the average separation for evaluating the solutions.

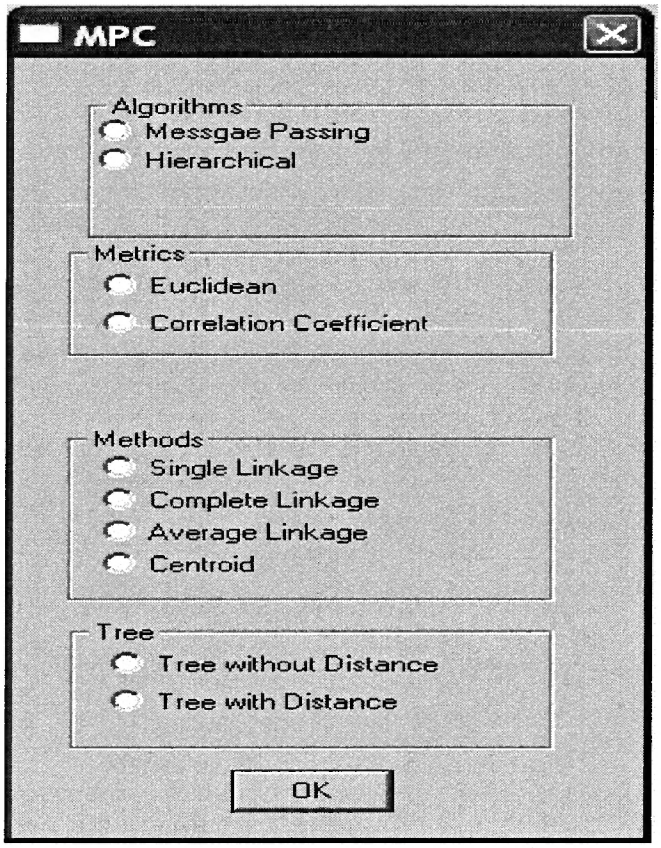


Figure 10. GUI of the Windows.

Figure 10 shows the graphic user interface for Windows version.

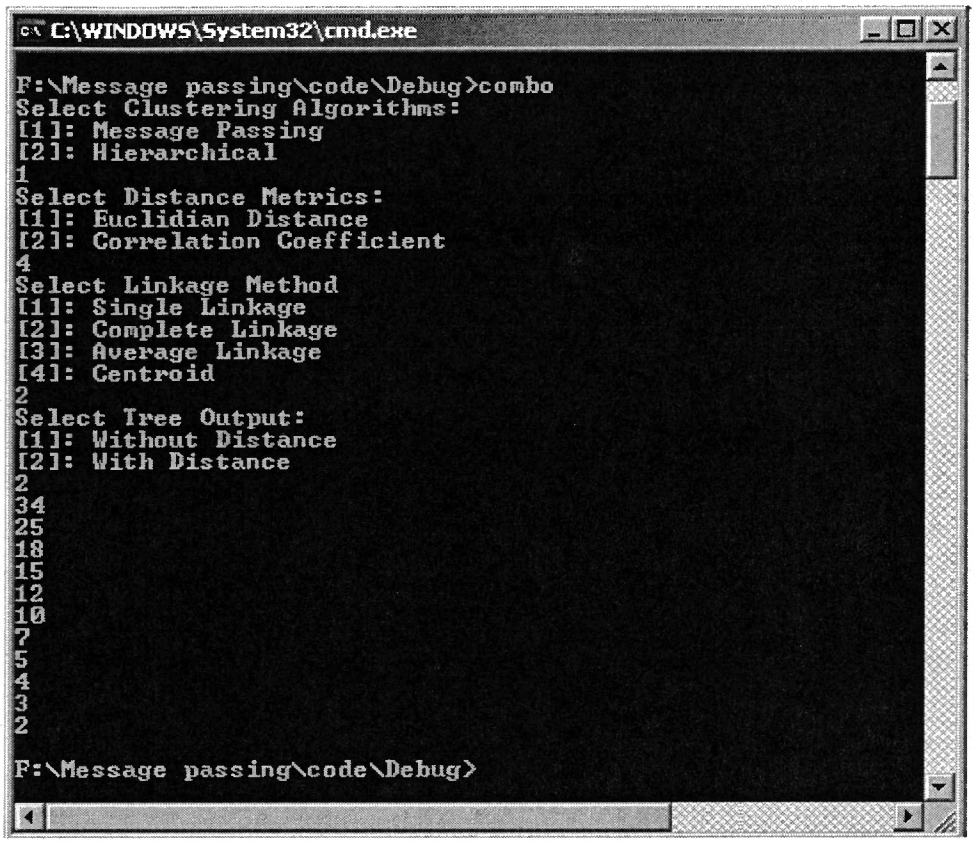


Figure 11. Command Line Interface of Dos version.

Figure 11 shows command line interface and an example output of the program when clustering 34 data objects. The steps of clustering processes are displayed.

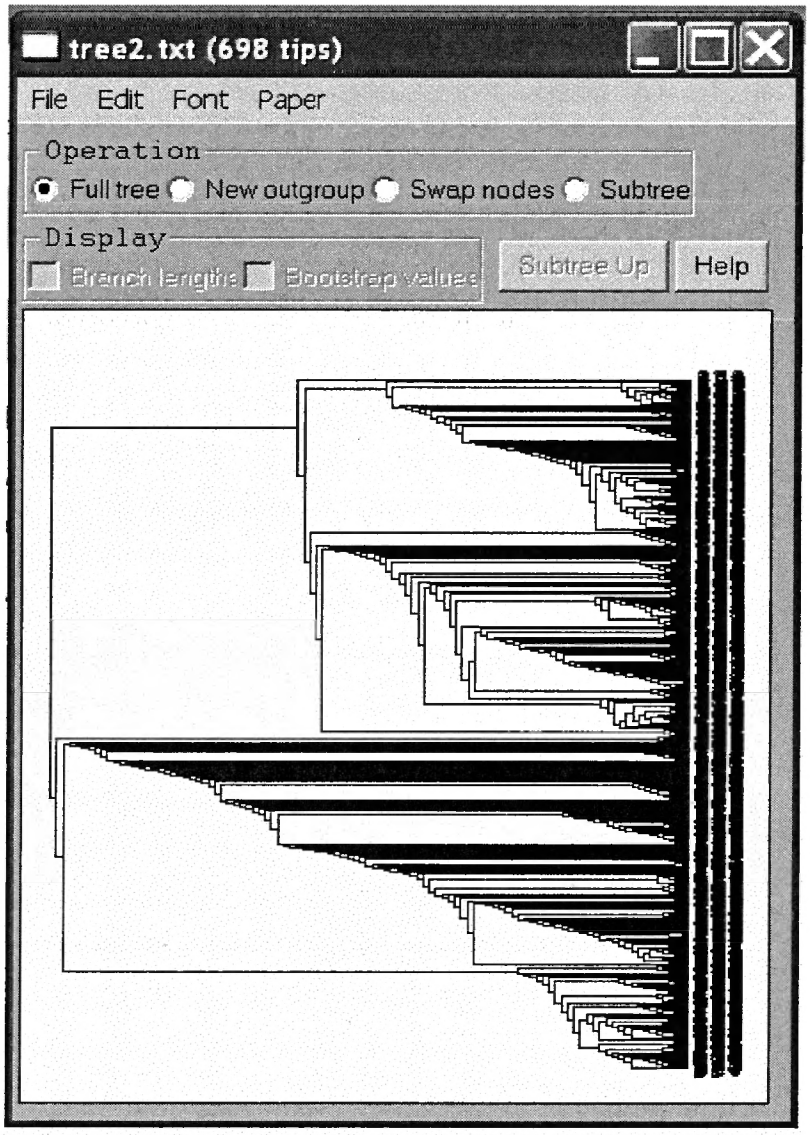


Figure 12. Clustering Dendrogram Outputted.

Several files are formed as the outputs: the 'tree' file is used as an input to a tree view software, NJplot [Perrière and Gouy 1996], to display the clustering dendrogram with/without branch lengths (Figure 12);

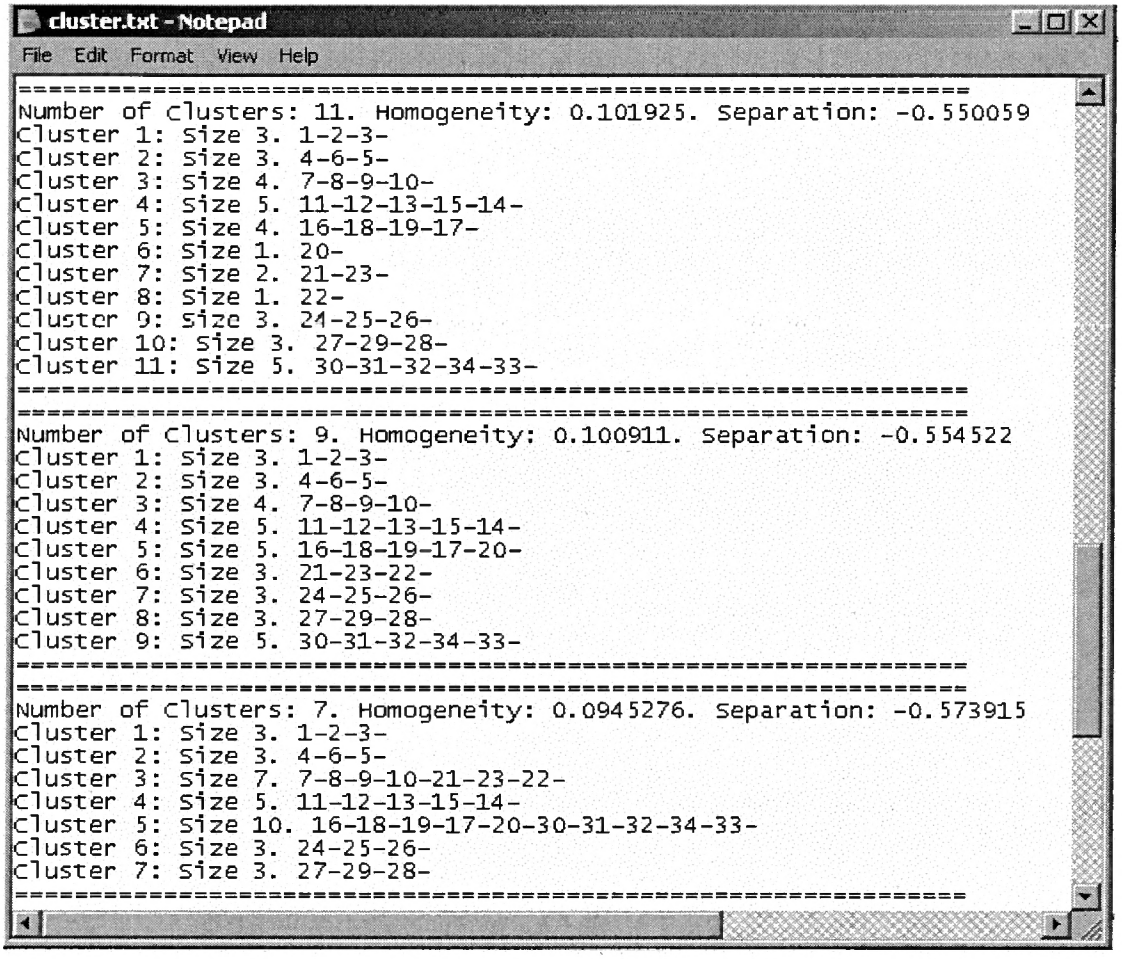


Figure 13. The Information Displayed in the Output File 'Cluster'.

The 'cluster' file (Figure 13) contains the information of clustering processes, including the number of clusters, the size of each cluster, the data objects in each cluster, and the average homogeneity and the average separation for evaluating the solutions.

Software and documentation for Windows™ and UNIX platforms can be downloaded freely at *http://bioinformatics.ist.unomaha.edu/~hgeng/*. The software is for noncommercial use only.

Chapter 5

RESULTS OF THREE CASE STUDIES

We test the validity of the MPC method in three ways. First, simulated microarray expression data was used to show that the proposed method has high accuracy and stability. Secondly, the real data representing the relatedness of DNA sequences of 34 strains of nine species of *Mycobacterium* was used to look at how well the results generated by the proposed method agree with the real phylogenies and to show the superiority of the proposed method over existing techniques such as the HC and neighbor joining (NJ) methods. Finally, the gene-expression data from the Stanford yeast cell cycle database was used to compare the solution generated from the proposed method to solutions from other clustering methods.

The centroid criterion was used in all of three experiments for two reasons: centroid linkage takes account of cluster structure while single or complete linkages do not, and centroid linkage assures that more often than not, both the intermediate and the final clustering results from MPC and HC are different.

Euclidean distance was used in the first two case studies with the simulated data and the real data from *Mycobacterium* sequences. In order to compare the solutions of MPC and solutions of other clustering methods, a correlation coefficient was used in the third case study with the gene expression data from the Stanford yeast cell-cycle database, to comply with the parameters chosen in Shamir *et al.*,.

5.1. SIMULATED DATA SETS

Since experimental gene-expression datasets typically have noise [Holter *et al.*, 2000] and their clusters may not completely interpret the underlying biological behavior which is derived from information other than gene-expression data, we would start our experiments with simulated data for which the classes are known.

An online simulator, eXPatGen [Michaud *et al.* 2003] (available at: http://www.che.udel.edu/eXPatGen/), was used in the first-step evaluation of the proposed method. Employing the user-defined inputs to the simulator, we can get dynamic mRNA profiles similar to those produced from microarray experiments from bacteria to human. Then the proposed method was applied in the analysis of the data and the results were directly compared to the initial input. If the data object is clustered in the same group in the output as in the original input, it's a hit; otherwise, it's considered as a miss. Figure 14 demonstrates the methodology of evaluating MPC with the simulated data.

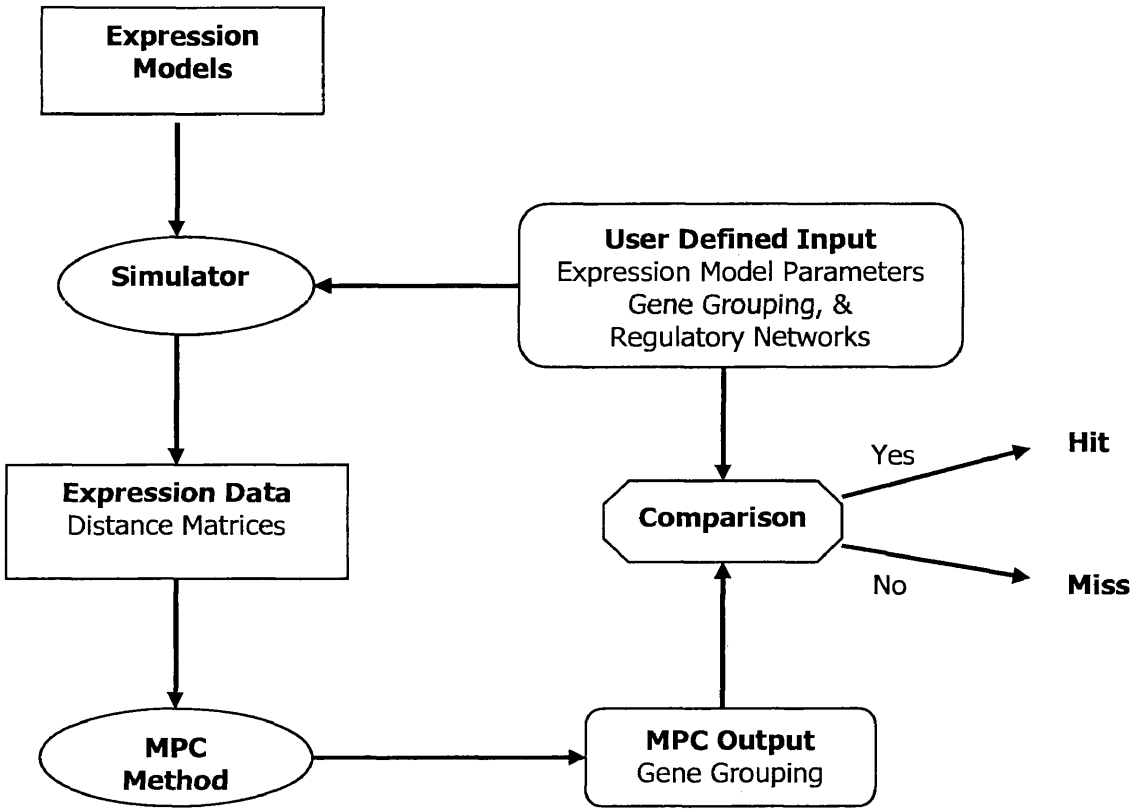


Figure 14. Evaluating MPC with the Simulated Data.

We will show one example of the simulated data sets, which contains 10 groups of genes, with 10 genes in each group, over 33 conditions which are a series of time points. By setting parameters, we can control which genes are induced or repressed by which genes, define the transcription dynamics for each gene, and hence determine the gene-expression patterns.

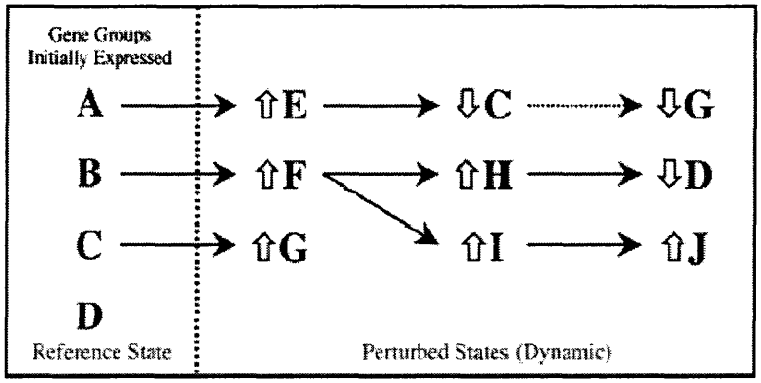


Figure 15. Regulatory Network for Our Example [Michaud *et al.* 2003].

The regulatory network for these gene groups (Figure 15) is used as part of the input to the simulator [Michaud *et al.* 2003].

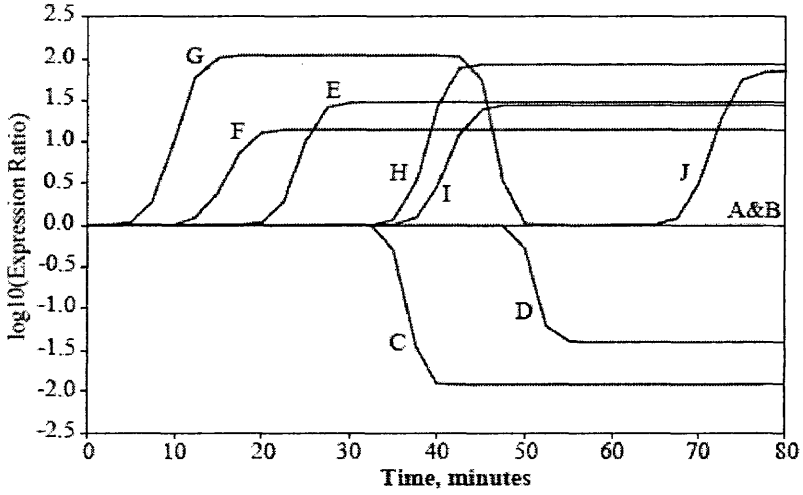


Figure 16. Dynamic Expression Profiles for Our Example [Michaud *et al.* 2003].

Figure 16 shows the dynamic expression profile of those 10 groups of genes [Michaud *et al.* 2003], which is the output from the simulator. Since A and B have the same dynamic expression profile, actually there are nine groups of genes with different expression patterns.

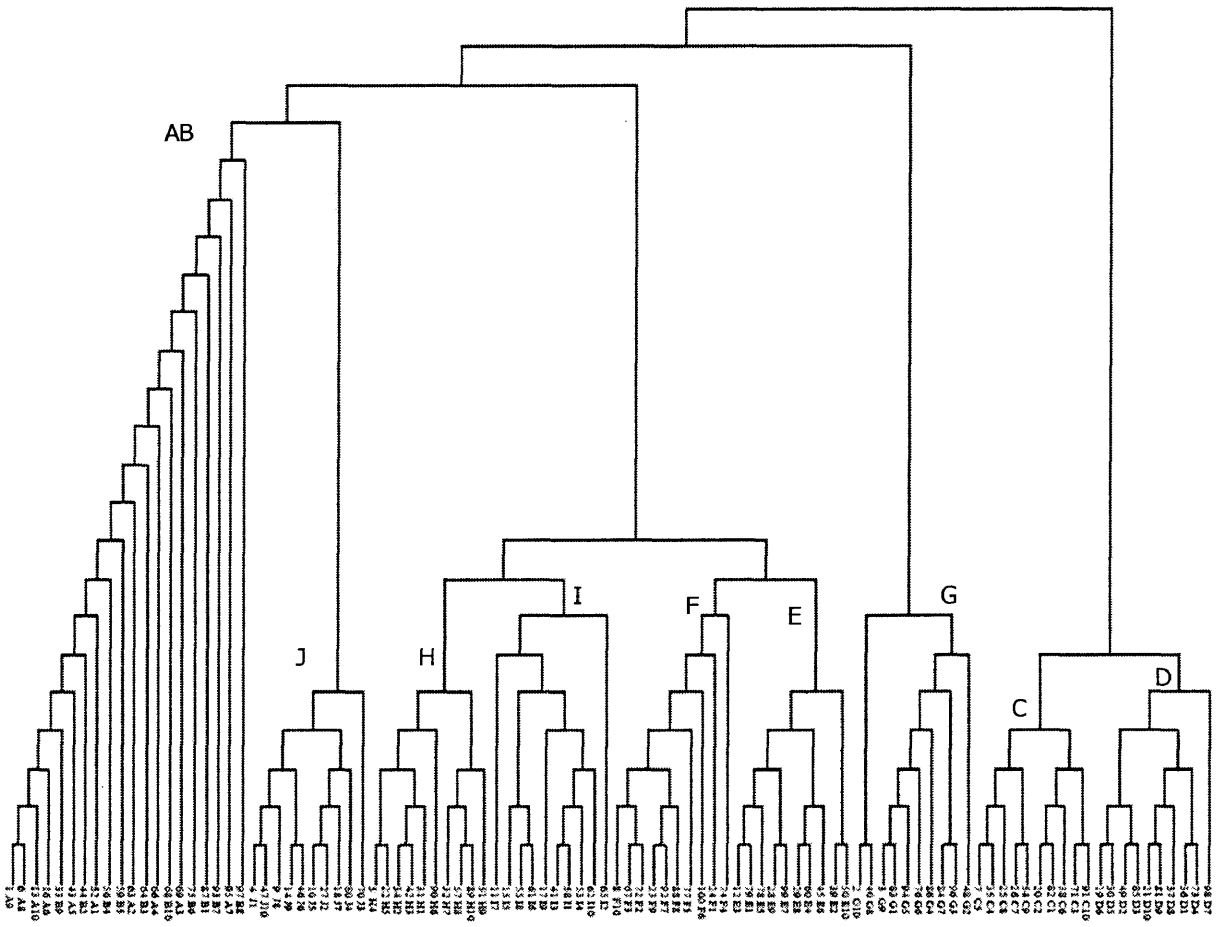


Figure 17. Clustered Display Using MPC Method.

Applying the proposed method to this data set, we found that the clustered display (Figure 17) includes exactly nine clusters, with each cluster presenting one gene group.

The data objects are clustered in the same group in the output as in the original input, so it's a hit.

In total, we tested 35 data sets. For each data set, 10 to 100 genes with dimension (a series of time points in the experiments) ranging from 20 to 40 were included. A 95% hit rate was achieved, in which 639 genes out of a total of 674 genes were correctly clustered. From this case study, we can see MPC has high accuracy and stability for simulated data sets, which encourages us to go to the next step: applying our method to real data sets.

5.2. REAL DATA SET _ *MYCOBACTERIUM* SEQUENCES

We have successfully applied the proposed method to constructing phylogenetic trees. In this case study, we would like to build the relationships among 34 strains of nine species of *Mycobacterium* using a traditional method, such as NJ, and the proposed method, and look at how well the solutions agree with the real phylogenies. The Relative Complexity Measure (RCM) algorithm [Out and Sayood 2003] that has successfully been implemented in the phylogenetic study of fungi [Bastola *et al.* 2004] was used in calculating the distances between all pairs of 34 strains including nine *Mycobacterium* species.

The description of strains is given in Table 5.

Table 5. Descriptions of Strains Used for Analysis.

Genus: *mycobacterium,* nine species, and 34 strains.

Species	Strain
chelonae	(ATCC 35752), (ATCC 19536), (DSM 43276)
fortiutum	(ATCC 49403), (ATCC 49404), (ATCC 43266), (ATCC 6841)
gordonae	(ATCC 14470), (ATCC 35756), (Bo 11340/99), (Bo 10681/99), (Bo 9411/99)
intracellulare	(ATCC 13950), (ATCC 35847), (ATCC 35770), (S 348), (S 350)
kansasii	(ATCC 12478), (S 221), (S 536), (S 233), (DSM 44431)
lavescens	(ATCC 14474), (DSM 43531), (ATCC 23033)
peregrinum	(ATCC 14467), (ATCC 700686), (S 254)
terrae	(ATCC 15755), (DSM 43541), (S 281)
xenopi	(ATCC 19250), (S 88), (S 91)

ATCC = American Type Culture Collection, Manassas, USA;

Bo = Elvira Richter, Nationales Referenzzentrum für Mykobakterien, Forschungszentrum Borstel, Germany;

S = Andres Roth, Institut für Mikrobiologie und Immunologie, Lungenklinik Heckeshorn, Berlin, Germany;

DMS = Deutsche Sammlung von Mikroorganismen und Zellkulturen GmbH, Braunschweig, Germany.

5.2.1. Neighbor Joining (NJ) method

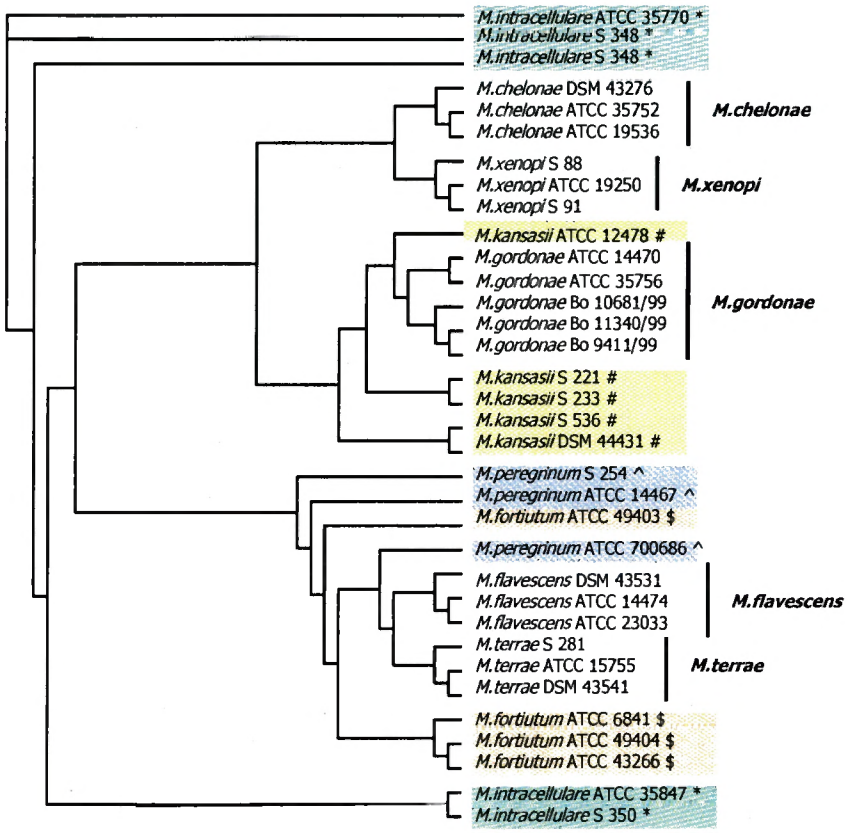


Figure 18. Phylogenetic Tree Constructed by NJ Method.

The distance-based tree (Figure 18) generated with the neighbor joining (NJ) program [Saitou and Nei 1987] in the PHYLIP package [Felsenstein 1989] demonstrates that only five out of a total of nine groups of species show biologically relevant clusters, in which all strains and only the strains (given in table 5) from the same species are grouped together, while the other four spices are mixed. Five strains from *Mycobacterium intracellulare* (marked by *) were not in the same group; three strains from

Mycobacterium peregrinum (marked by ^) were not in the same group; the strain ATCC 49403 from *Mycobacterium fortuitum* (marked by $) was not grouped with the other three; and the strain ATCC 12478 from *Mycobacterium kansasii* (marked by #) was misclassified into *Mycobacterium gordonae*.

5.2.2. Message Passing Clustering (MPC) method

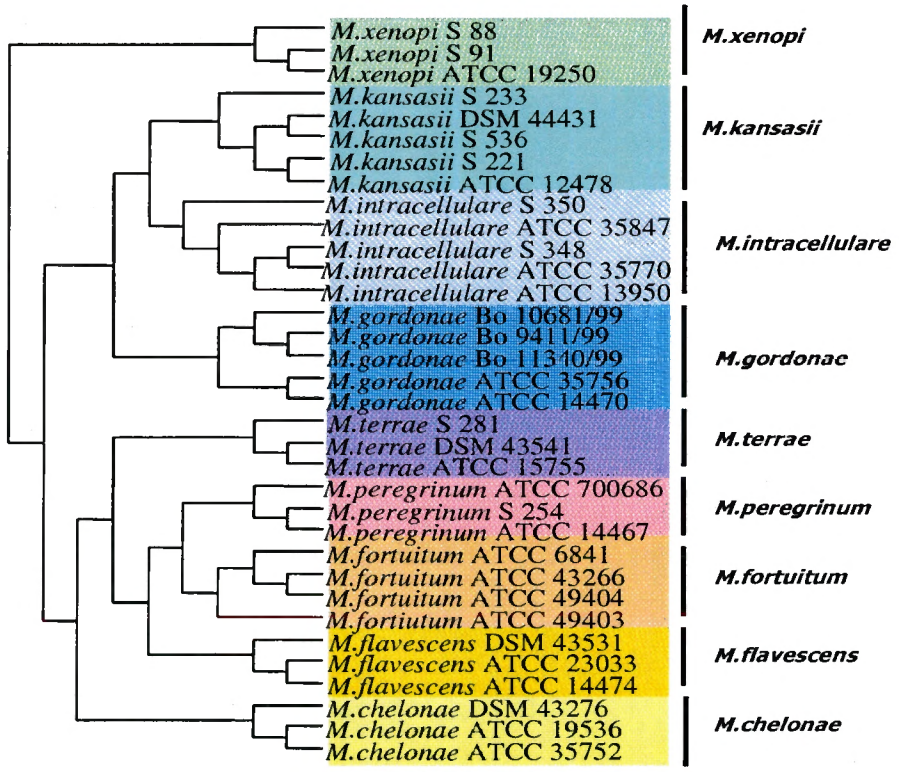


Figure 19. Phylogenetic Tree Constructed by MPC Method.

The phylogenetic tree obtained by the MPC method (see Figure 19) shows a better picture of the *Mycobacterium* species. It generates exactly nine clusters, representing multiple strains from nine different species.

5.2.3. Hierarchical Clustering (HC) method

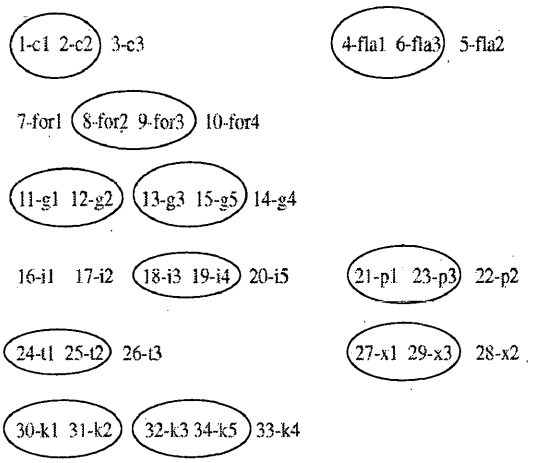
The analysis of the same data set with the HC method shows different topology compared with the MPC method and incorrect positioning of those strains, because HC is a greedy algorithm which considers only the global structure but does not honor local structure.

Table 6. Indices of 34 Strains of Nine *Mycobacterium* Species.

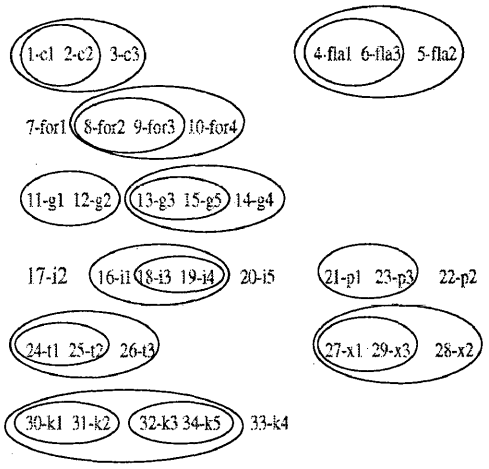
Species	Index	Strains
M.chelonae	1~3	c1,c2, c3
M.flavescens	4~6	fla1, fla2, fla3
M.fortiutum	7~10	for1, for2, for3, for4
M.gordonae	11~15	g1, g2, g3, g4, g5
M.intracellulare	16~20	i1, i2, i3, i4, i5
M.peregrinum	21~23	p1, p2, p3
M.terrae	24~26	t1, t2, t3
M.xenopi	27~29	x1, x2, x3
M.kansasii	30~34	k1, k2, k3, k4, k5

In order to demonstrate the clustering processes easily and clearly, we assign an index and an abbreviated name to each strain, as shown in Table 6.

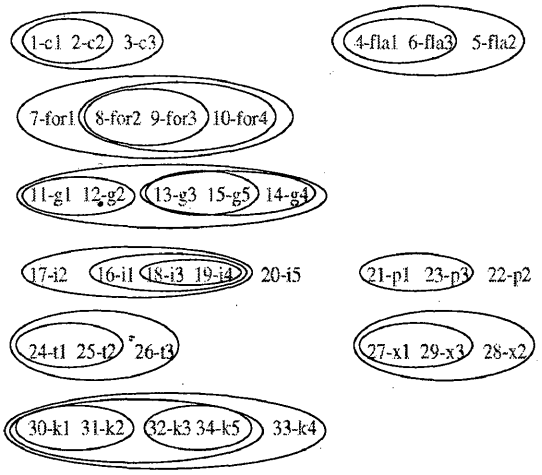
In the following figures we will show the clustering processes of the MPC method and the HC method and illustrate the difference between them.



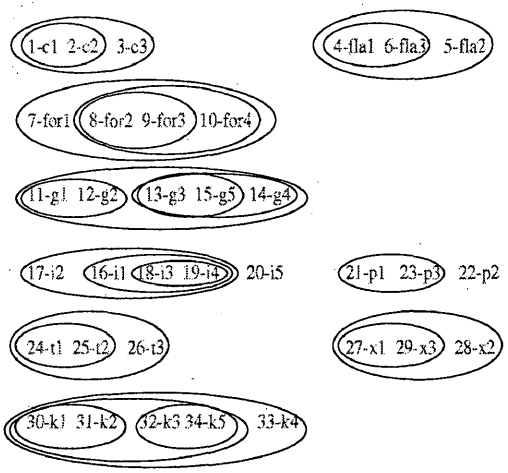
(a). MPC loop 1: 34 data objects→23 clusters.



(b). MPC loop 2: 23 → 15 clusters.



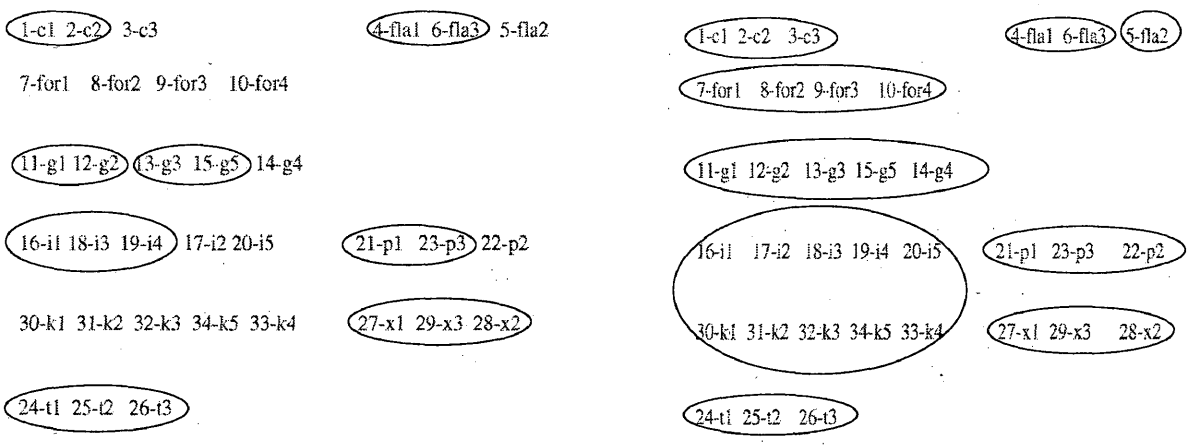
(c). MPC loop 3: 15 → 11 clusters.



(d). MPC loop 4: 11 → 9 clusters.

Figure 20. Clustering Processes of MPC.

Our goal is to cluster 34 strains into nine groups since we have already known there are nine different species. Figure 20 illustrates the clustering processes of MPC. In each step, several clusters are formed spontaneously so that clusters are generated in parallel. Starting from the initial objects, only four steps are needed to get to the final solution. Also clusters generated in each step are evenly distributed among nine species because both local and global structures are honored in MPC. Other advantages of MPC in this case study include reducing the number of singletons and giving some suggestions about the number of clusters since not every number can be reached.



(a). HC loop 11: 34 data objects→23 clusters. (b). HC loop 25: 9 clusters.

Figure 21. Clustering Processes of HC.

From the clustering processes of HC (Figure 21), we have three observations. First, in only one step MPC produces 23 clusters (Figure 20 (a)), while 11 steps are needed in HC (Figure 21 (a)). In only four steps MPC finds the final solution, nine clusters, while 25 steps are needed in HC. This is because MPC generates clusters in parallel but HC generates clusters sequentially. Second, it is more likely that the solution given by HC contains more singletons (such as in Figure 21 (a)) compared to that given by MPC, because HC considers only global structure and merges the two clusters with the globally lowest distance at that point but doesn't care about the relationship in local area. Third, the final clusters given by HC are not accordant with the real phylogenies. As shown in Figure 21 (b), two species, intracellulare and kansasii, are grouped together as one species; the species flavescens is separated as two species.

This case study shows the superiority of the MPC method to the NJ and HC methods. From evolutionary history, it's common that different branches diverge at the same or close time point. Evolution of species happens in parallel but not sequentially. The key reason for MPC generating more biologically relevant topologies than those generated by HC is that MPC describes parallel biological processes while HC can present only sequential relationships.

5.3. REAL DATA SET _ GENE EXPRESSION DATA

The Stanford yeast cell-cycle database (available at: http://cellcycle-www.stanford.edu) has been widely used and well-analyzed by researchers. This database contains the gene expression levels of yeast ORFs (Open Reading Frames) over

79 conditions. Spellman *et al.* [Spellman *et al.*1998] identified in the original data 800 genes that are cell-cycle regulated, and further, Shamir et al. [Sharan and Shamir 2000] selected 698 of those 800 genes, which have no missing entries, over 72 conditions, to evaluate different clustering algorithms: K-mean [MacQueen 1965, Ball and Hall 1967, Herwig *et al.* 1999], CAST [Ben-Dor *et al.* 1999], SOM [Kohonen 1997, Tamayo *et al.* 1999], CLICK [Sharan and Shamir 2000], and 'True' [Spellman *et al.*1998].

In order to evaluate the MPC method compared with other popular clustering methods, we analyzed this 698×72 data set using the MPC method and set up the experiment using the same parameters as those chosen in Shamir *et al.* [Sharan and Shamir 2000]. Here, correlation coefficient similarity metric and centroid linkage distance criterion were used. Based on the analysis conducted by Spellman *et al.* [Spellman *et al.*1998], we expect to find in the data five main clusters: G1-peaking genes, S-peaking genes, G2-peaking genes, M-peaking genes, and M/G1-peaking genes.

In Figure 22, we show the dynamics of homogeneity and separation in the entire clustering process. The turning point in the number of clusters is eight. The magnified graph is shown in Figure 23. When looking at the size of each cluster, we found that there exist two singletons and one small cluster containing only six genes when the number of clusters is eight. This suggests the final number of clusters should be five after discarding two singletons and one small cluster. This is in accordance with the number of clusters that we expected to find.

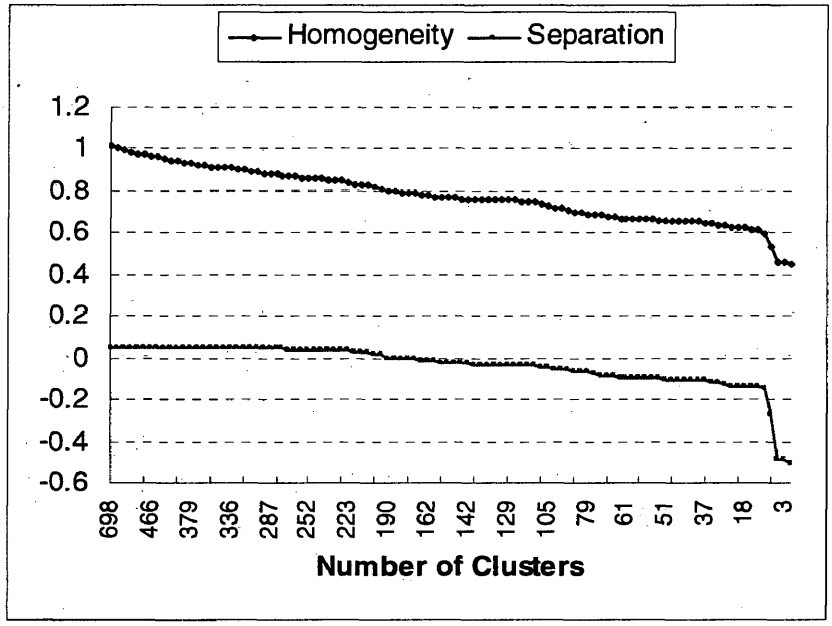


Figure 22. Dynamic Homogeneity and Separation Values in the Entire Clustering Process.

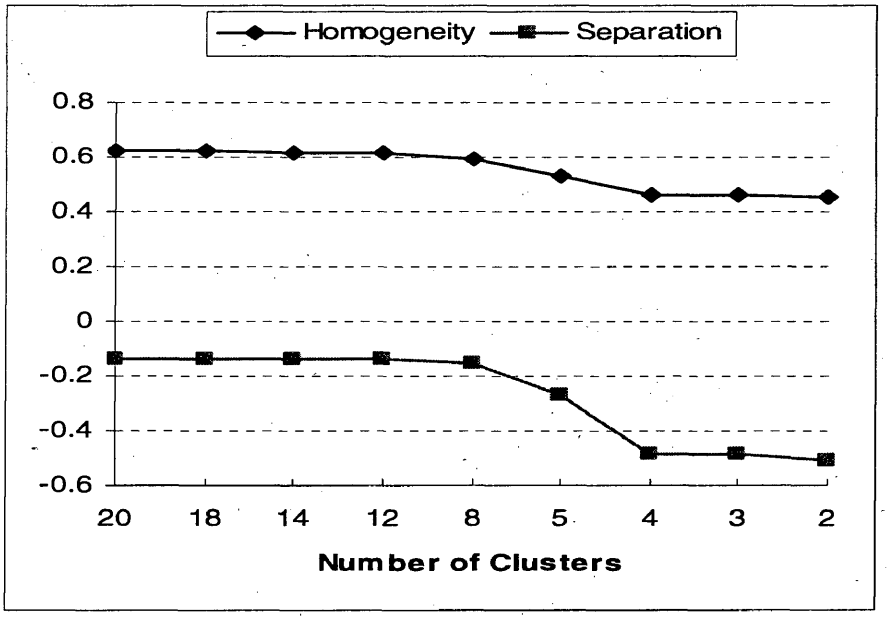


Figure 23. Magnified Graph for Homogeneity and Separation Values.

A potential value of the figure presenting the dynamic homogeneity and separation in the entire clustering process is its ability to suggest the 'true' number of clusters for the data by observing significant changes of homogeneity and separation and considering the size of clusters.

Table 6 shows the solutions produced by all programs and their homogeneity and separation parameters. GeneCluster is a gene expression clustering software which uses the SOM algorithm [Tamayo *et al.* 1999]. The so-called 'True' clustering is obtained manually by inspecting the expression patterns and comparing to the literature [Spellman *et al.*1998].

Table 7. Comparison of all clustering solutions.

Program	#Clusters	Homogeneity	Separation
K-Means	49	0.629	0.086
CAST	5	0.6	-0.146
GeneCluster	6	0.617	-0.073
CLICK	6	0.656	-0.098
'True'	5	0.572	-0.133
MPC	5	0.592738	-0.152215

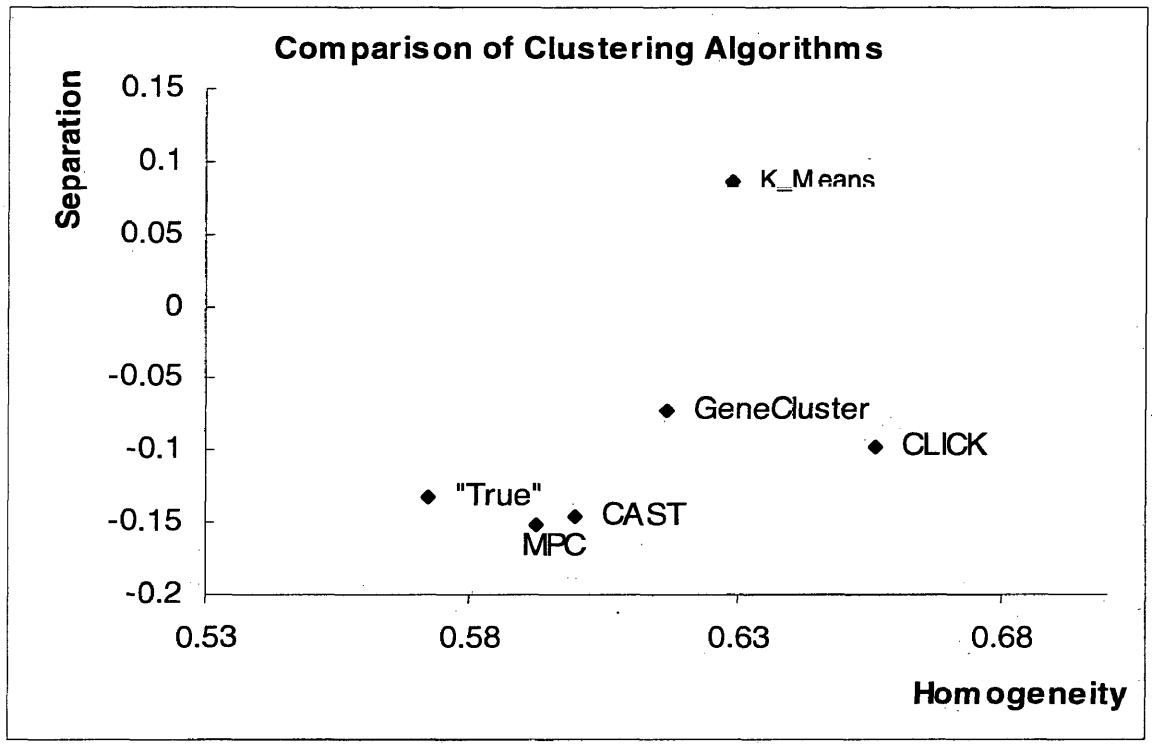


Figure 24. A Comparison of Homogeneity and Separation Values for all Solutions.

We compare homogeneity and separation values for all solutions in Figure 24. The ideal point is the lower right corner where both are good. However, homogeneity and separation are two conflicting parameters. Improvement of one will deteriorate the other. Therefore, for a given data set, all reasonable clustering algorithms will produce results lying in or near a curve. From Figure 12, the partition given by MPC is the closest to the 'true' partition among all solutions.

Chapter 6

CONCLUSIONS AND DISCUSSION

6.1. CONCLUSIONS

We proposed a new clustering algorithm which employs the concept of message passing. By taking advantage of the communication among data objects and by taking into account both local and global structure, MPC can describe parallel and spontaneous biological processes more precisely and produce more accurate clustering solutions.

While it may appear that MPC and HC produce similar outputs, we show that more often than not, the output clusters of the two approaches are different in favor of the MPC method. Particularly, if the centroid linkage is used as the distance criterion, the results obtained by MPC and HC are often different.

We have proved that MPC shares similarity with HC but offers significantly improved performance. Results from the three case studies assure the validity of the MPC method. In the first one, a 95% hit rate was achieved for the simulated gene expression data. In the second study, biologically relevant topology for different *Micobacterium* species was reflected in the phylogenetic tree. In the third case study, MPC beats all other algorithms by providing a partition that is closest to the 'true' solution.

6.2. DISCUSSION

In this thesis we addressed the intrinsic problems of HC by proposing a new algorithm, MPC, with which parallel processes can be well-modeled and captured. We predicted MPC should outperform most clustering algorithms when applied to biological data sets. The experimental results confirmed our hypotheses and warranted a promising future for developments and applications.

We applied the MPC method to biological data exclusively in this thesis. However, MPC is a general purpose clustering algorithm which can be applied to a wide range of scenarios such as marketing, city planning, battlefield controlling, WWW classification, etc. MPC is especially suitable for the data generated from parallel processes. In fact, most real-world data are generated in a parallel fashion. In our future plan, MPC will be applied to data from a variety of domains.

A subtle point about MPC is: not every number K may be exactly reached because multiple clusters may be merged in parallel in a message sending/receiving round. This is not necessarily bad because we can modify the pseudocode by ranking the ready-to-merge clusters and merge only a portion of clusters with the high similarities. In fact, the number K can tell something about the data. An unreachable K suggests that K is not a good number for the data. As we described in the previous section, we can predict the 'true' number of clusters for a given data set by monitoring the homogeneity and separation values during the clustering process.

Another advantage of MPC is its flexible structure which allows future development under the same logic frame. We could change the message type and the way

the message is handled to deal with more challenging situations. The following picture shows a real-world example how clusters can be formed based on different attributes.

This picture was taken at IEEE CSB2004 conference at the Stanford University. If we looked at each table as a cluster, then we found that people around some table are grouped based on the university: for example, around the closest table, the people are all from the University of Nebraska at Omaha; while people around another table may be grouped based on the common research interest or based on the country they are from, because they all speak the same language. We can cluster the same data set based on different data attributes.

As another example, if the attributes consist of a mixture of data types, such as nominal, ordinal, and scalar data, the distance between two data objects is not well-

defined. This problem is algorithm-independent, but it can be handled by modifying MPC. In this case, we could send multiple messages according to the data types of the attributes for each data object. This way, the messages are handled independently and clusters are formed based on all messages the objects sent and received. This more complex version of MPC is under development.

A common complaint about HC and MPC is that they can never undo what is previously done. We can extend MPC to a probabilistic MPC to overcome this problem. We could adjust the sending and receiving procedure by sending messages from each cluster to multiple clusters and assigning merging probabilities to those messages. In the receiver side, we recalculate the probability of each node which was already in the cluster and kick out the nodes which no longer have good probability, so that the irreversible problem in HC and MPC is solved. With the probability assigned to each message, MPC can also be made fussy clustering, which means each object could belong to more than one cluster.

6.3. Future Work

In our future plan, we would like to:

- Apply MPC to other data domains.
- Extend MPC to a probabilistic version.
- Change the message type and the way the massage is handled to deal with more challenging situations.

REFERENCE

1) [Alter *et al.* 2002], O. Alter, P.O. Brown, and D. Botstein, "Single value decomposition for genome-wide expression data proessing and modeling", *Proc. Natl Acad. Sci. USA*, 2000, vol. 97, pp. 10101-10106.

2) [Ball and Hall 1967] G. Ball and D. Hall, "A clustering technique for summarizing multivariate data", *Behaviorial Sciences*, 1967, 12(2), pp. 153-155.

3) [Bastola *et al.* 2004] D.R. Bastola, H.H. Out, S.E. Doukas, K. Sayood, S.H. Hinrichs, and P.C. Iwen, "Utilization of the relative complexity measure to construct a phylogenetic tree for fungi", *Mycol Res,* 2004, vol 108, pp.117-125.

4) [Ben-Dor *et al.* 1999]. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering Gene Expression Patterns", *J. Comput. Biol.*, 1999, vol 6, pp. 281-297.

5) [Bittner *et al.* 2000]; M. Bittner *et al.*, "Molecular classification of cutaneows malignant melanoma by gene expression profiling", *Nature*, 2000, vol. 426, pp. 536-540.

6) [Brown *et al.* 2000] M.P. Brown *et al.,* "Knowledge-based analysis of microarray gene expression data by using support vector machines", *Proc. Natl Acad. Sci. USA*, 2000, vol. 97, pp. 262-267.

7) [Butte *et al.* 2001] A.J. Butte, J. Ye, G. Niederfellner, K. Rett, H.U. H'firing, M.F.White, and I. S. Kohane, "Determining Significant Fold Differences in Gene Expression Analysis", *Pacific Symposium on Biocomputing*, 2001, vol. 6, pp. 6-17.

8) [Cho *et al.* 1998] R.J. Cho, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockhart, and R.W. Davis, "A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle", *Molecular Cell*, 1998, vol. 2, pp. 65-73.

9) [Churchill 2002] F.A. Churchill, "Fundamentals of Experimental Design for cDNA Microarrays", *Nature genetics suppl.*, 2002, vol. 32, pp. 490-495.

10) [De Smet *et al.* 2002], F. De Smet *et al.*, "Adaptive quality-based clustering of gene expression profiles", *Bioinformatics*, 2002, vol. 18, pp. 735-76.

11) [Dudoit *et al.* 2000] S. Dudoit, J. Fridlyand, and T.P. Speed, "Comparison of discrimination methods for the classification of tumors using gene expression data". Technical Report 576. (Department of Statistics, University of California

at Berkeley, Berkeley, CA, 2000).

12) [Eisen *et al.,* 1998] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns", *Proc Natl Acad Sci, U S A*, 1998 Dec. vol. 95, pp. 14863-14868.

13) [Ermolaeva *et al.* 1998] O. Ermolaeva *et al.*, "Data Management and Analysis for Gene Expression Arrays", *nature genetics,* 1998, vol. 20.

14) [Everitt 1993] B. Everitt, *Cluster Alalysis,* Halsted Press, New York, 1993.

15) [Felsenstein 1989] J. Felsenstein, PHYLIP (Phylogeny Inference Package), *Cladistics*, 1989, vol. 5, pp. 164-166.

16) [Fraley and Raftery 2002]. C. Fraley and A.E. Raftery, "Model-based clustering, discriminant analysis, and density estimation", *J. Amer. Stat. Assoc.,* 2002, vol. 97, pp. 611-631.

17) [Fuhrman *et al.* 1998] X. Wen, S. Fuhrman, G.S. Michaels, D.B. Carr, S. Smith, J.L. Barker, and R. Somogyi, "Large-scale temporal gene expression mapping of central nervous system development", *Proc. Natl. Acad. Sci. USA,* 1998, vol. 95, pp. 334-339.

18) [Furey *et al.* 2000], T.S. Furey *et al.,* "Support vector machine classification and validation of cancer tissue samples using microarray expression data", Bioinformatics, 2000, vol. 16, pp. 906-914.

19) [Gullans 2000] Steven R. Gullans (2000) "Of microarrays and meandering data points". *Nature genetics,* 2002, vol. 26.

20) [Hartemink *et al.* 2001], A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, and R.A Young, "Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks". *Pac. Symp. Biocomput.*, 2001, pp. 422–433.

21) [Hartemink *et al.* 2002], A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, and R.A. Young, "Combining location and expression data for principled discovery of genetic regulatory network models" *Pac. Symp. Biocomput.,* 2002, pp. 437–449.

22) [Hastie *et al.* 2000], T. Hastie *et al.*, "'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns", *Genome Biol.,* 2000, vol. 1, research 0003.

23) [Hedenfalk *et al.* 2001], I. Hedenfalk *et al.* "Gene-expression profiles in

hereditary breast cancer". *N. Engl. J. Med.*, 2001, vol. 344, pp. 539–548.

24) [Herwig *et al.* 1999] R. Herwig, A. J. Poustka, C. Meuller, H. Lehrach, and J. O'Brien, "Large-scale clustering of cDNA-fingerprinting data", *Genome Research*, November 1999, 9(11), pp. 1093-1105.

25) [Heyer *et al.* 1999], L,J. Heyer, S. Kruglyak, and S. Yooseph, "Exploring expression data: identification and analysis of coexpressed gene", *Genome Res.*, 19999, vol. 9, pp. 1106-1115.

26) [Holter *et al.*, 2000], M.S. Holter, M. Mitra, A. Maritan, M. Cieplak, J.T. Banavar, and N.V. Fedoroff, "Fundamental Patterns Underlying Gene Expression Profiles: Simplicity from Complexity", *PNAS*, 2000, vol. 97, pp. 8409-8414.

27) [Jain and Dubes 1998], A.K. Jain, and R.C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1998.

28) [Khan *et al.* 1998], J. Khan *et al.*, "Gene expression profiling of alveolar rhabdomyosarcoma with cDNA microarrays", *Cancer Res.*, 1998, vol. 58, pp. 5009-5013.

29) [Kohonen 1997] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, 1997.

30) [Lance and Williams 1967] G.N. Lance and W.T. Williams, "A general theory of classification sorting strategies", *the computer journal*, 1967, vol. 9, pp. 373-380.

31) [Landgrebe *et al.* 2002], J. Landgrebe, W. Wurst and G. Welzl, "Permutation-validated principal components analysis of microarray data", *Genome Biol. 3*, 2002, research 0019.

32) [MacQueen 1965]J. MacQueen, "Some methods for classication and analysis of multivariate observations", In Proceedings of *the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1965, pp. 281-297.

33) [Michaud *et al.* 2003] D.J. Michaud, A.G. Marsh, and P.S. Dhurjati, "eXPatGen: generating dynamic expression patterns for the systematic evaluation of analytical methods", *Bioinformatics,* 2003, vol. 19 no. 9, pp. 1140-1146.

34) [Mitchell 1997], T.M. Mitchell, *Machine Learning*, 414 (WCB McGraw-Hill, Boston, 1997).

35) [Ooi and Tan 2003] C.H.Ooi, and P. Tan, "Genetic Algorithms Applied to Multi-class Prediction for the Analysis of Gene Expression Data", *Bioinformatics*, 2003, vol. 19, pp. 37-44.

36) [Out and Sayood 2003] H. H. Out and K. Sayood, "A new sequence distance measure for phylogenetic tree construction", *Bioinformatics*, 2003, vol. 19, pp. 1-9.

37) [Pe'er *et al.* 2001] D. Pe'er, A. Regev, G. Elidan, and N. Friedman, "Inferring subnetworks from perturbed expression profiles", *Bioinformatics,* 2001, vol. 17 Suppl. 1, S215–S224.

38) [Perrière and Gouy 1996] G. Perrière and M. Gouy, "WWW-Query: An on-line retrieval system for biological sequence banks", *Biochimie*, 1996, vol. 78, pp. 364-369.

39) [Quackenbush 2002] John Quackenbush, "Microarray Data Normalization and Transformation", *Nature genetics suppl.*, 2002, vol. 32, pp. 496-501.

40) [Raychaudhuri *et al.* 2000], S. Raychaudhuri, J.M. Stuart, and R.B. Altman, "Principal components analysis to summarize microarray experiments: application to sporulation time series", *Pac. Symp. Biocomput.* 2000, pp. 455-466.

41) [Saitou and Nei 1987] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees", *Mol. Biol. Evol.*, 1987, vol. 4, pp. 406-425.

42) [Segal *et al.* 2001] E. Segal, B. Taskar, A. Gasch, N. Friedman, and F. Koller, "Rich probabilistic models for gene expression". *Bioinformatics*, 2001, vol. 17, Suppl 1, S243–S252.

43) [Sharan and Shamir 2000]. R. Shamir, and R. Sharan, "Algorithmic Approaches to Clustering Gene Expression Data", *Current Topics in Computational Molecular Biology*, pp. 269-300, MIT Press, 2002.

44) [Sharan and Shamir 2000]. R. Sharan and R. Shamir, "CLICK: A clustering algorithm with applications to gene expression analysis", In Proceedings *of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB),* 2000, pp. 307-316.

45) [Slonim, 2002]. D.K. Slonim, "From Patterns to Pathways: Gene Expression Data Analysis Comes of Age", *Nature genetics suppl.* 2002, vol. 32, pp.502-508.

46) [Spellman *et. al.*1998]. P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher, "Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization", *Molecular Biology of the Cell*, 1998, vol. 9, pp. 3273-3297.

47) [Tamayo *et al.* 1999] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T.R. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation", *PNAS*, 1999, vol. 96, pp. 2907-2912.

48) [Tibshirani 2002]. R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression", *Proc. Natl Acad. Sci. USA*, 2002, vol. 99, pp. 6567–6572.

49) [Golub *et al.* 1999]. T.R. Golub *et al.*, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring", *Science*, 1999, vol. 286, pp. 531–537.

50) [Yeung *et al.* 2001]. K.Y. Yeung, C. Fraley, A. Murua, A.E. Raftery, and W.L. Ruzzo, "Model-based clustering and data transformations for gene expression data", *Bioinformatics,* 2001, vol. 17, pp. 977-987.

51) [Yeung *et al.* 2001]. K.Y. Yeung, D.R. Haynor and W.L. Ruzzo, "Validating clustering for gene expression data", *Bioinformatics,* 2001, vol. 17 no. 4, pp. 309-318.

52) [Yeung *et al.* 2002] K.Y. Yeung, M.T. Barrett, J, Delrow, P.L. Blount, L. Hsu, W.L. Tuzzo, B.J. Reid, and P.S. Rabinovitch, "Expression analysis of Barrett's epithelium and normal gastrointestinal tissues", 2000.

53) [Yoo *et al.* 2002] C. Yoo, V. Thorsson, and G.F. Cooper, "Discovery of causal relationships in a gene-regulation pathway from a mixture of experimental and observational DNA microarray data". *Pac. Symp. Biocomput.*, 2002, pp. 498–509.