Algorithms for hierarchical clustering of gene expression data

by

Srikanth Komarina

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
Srinivas Aluru, Major Professor
Xiaoqiu Huang
Srikanta Tirthapura

Iowa State University

Ames, Iowa

2004

Graduate College
Iowa State University

This is to certify that the master's thesis of

Srikanth Komarina

has met the thesis requirements of Iowa State University

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Genes are parts of the genome which encode for proteins in an organism. Proteins play an important part in many biological processes in any organism. Measuring expression level of a gene helps biologists estimate the amount of protein produced by that gene. Mircoarrays can be used to measure the expression levels of thousands of genes in a single experiment. Using additional techniques such as clustering, various correlations among genes of interest can be found. The most commonly used clustering technique for microarray data analysis is hierarchical clustering. Various metrics such as euclidean, manhattan, pearson correlation coefficient have been used to measure (dis)similarity between genes. A commonly used software for hierarchical clustering based on pearson correlation coefficient takes $O(N^3)$ for clustering N genes, even though there are algorithms which can reduce the runtime to $O(N^2)$. In this thesis, we show how the runtime can be reduced to $O(Nlog\ N)$ by using a geometric interpretation of the pearson correlation coefficient and show that it is optimal.

CHAPTER 1. INTRODUCTION

Cells are the fundamental working units of every living system. All instructions needed to perform various activities are contained within a double stranded molecule called *Deoxyribonucleic acid* (*DNA*). DNA is composed of four nucleotides (bases), denoted by A, C, G and T. The term *genome* refers to an organism's complete set of DNA. The *genes*, parts of the DNA, are the functional units of the genome. Genes carry instructions for making proteins in the cell. Proteins play an important part in many of biological processes that take place in the cells. *Gene expression* refers to a two-step process that begins with a process called *transcription* which transforms gene into another molecule called *Ribonucleic acid* (*RNA*). The RNA is a single stranded nucleic acid sequence made up of nucleotides A, C, G and U. The RNA molecule is then translated into the corresponding protein and this process is referred to as *translation*. The expression level of a gene directly impacts the amount of protein produced from it and could vary drastically based on the underlying biological conditions. Biologists are interested in measuring the expression levels of various genes in an organism under different biological conditions. The experiments performed to understand biological phenomenon at the molecular level have traditionally involved studying one gene per experiment. This is highly time consuming and becomes infeasible when there are a large number of genes to be simultaneously studied. For example humans contain about 30,000 genes [26]. Hence, the need arises to study expression levels of thousands of genes simultaneously. The advent of microarray technology has made it possible to study the entire genome of an organism using a single chip [6].

Microarrays are used to measure the expression levels of thousands of genes simultaneously. An array is an orderly arrangement of samples. It provides a medium for matching known and unknown DNA samples based on base pairing rules. An array experiment can use microplates or standard blotting membranes, and can be created by hand or use robotics to deposit the sample generally on glass but sometimes on nylon substrates. These arrays are described as *macroarrays* or *microarrays*, the difference being the size of sample spots. Macroarrays contain sample spot sizes of about 300 microns or larger and can be easily imaged by existing gel or blot scanners. The sample spot sizes in microarray are typically less than 200 microns in diameter and these arrays usually contain thousands of spots. Microarrays require specialized robotics and imaging equipment that generally are not commercially available as a complete system. Potential applications of microarrays include identification of complex genetic diseases, drug discovery and toxicology studies, pathogen analysis and differing expression of genes over time, between tissues, and disease states.

Microarray technology is based on the ability of complimentary base pairing of the nucleic acid. Probe *DNA* (or *RNA*) strands are spotted onto the chip. Target *DNA* mixture (usually *cDNA*, corresponding to the genes) is flooded onto the chip to allow base pairing under conditions such that only highly complimentary sequences will remain bound to their specific partners. Technology differs in how DNA sequences are laid down (spotting vs photolithography) and the length of *DNA* sequences that are laid down (complete sequences or a series of fragments). To detect the *cDNA*'s bound to the microarray they are labeled with fluorescent dyes. Dyes used are CY3 (green) and CY5 (red). Microarray is then excited by laser light at two different frequencies, to measure the expression level at each spot. The expression level is the ratio of the brightness of the hybridized *RNA* of one sample versus another, i.e. CY3/CY5. If a background intensity for each color is measured and controlled for, the ratio becomes $\frac{CY3-CY3\ background}{CY5-CY5\ background}$. This provides a measure of relative abundance of the RNA from one sample to another and hence the relative expression level of the corresponding gene in the samples. Once the expression levels of each of the genes under various experimental conditions are measured, the data must be analyzed to derive inferences. We are interested in *patterns* of expression across multiple genes and experiments. To detect such patterns, additional methods such as clustering are needed.

A simple clustering method would be to identify all genes whose expression level has changed and those which remain unchanged, thus resulting in two clusters. At the next level of data analysis we would like to detect the covariance among genes i.e. whether there exists multiple gene patterns, clusters of genes that share the same behavior. Clustering methods can be generalized into two general classes, supervised and unsupervised clustering [15]. In supervised clustering objects are classified with respect to known information. In unsupervised clustering no predefined information is available. Since there is little *a priori* knowledge of the complete repertoire of expected gene expression patterns for any condition, unsupervised clustering methods or hybrid (unsupervised followed by a supervised) methods are preferred. Various clustering techniques have been applied to the microarray data [1, 5, 7, 10, 23, 24, 25]. It is not possible to single out a *best method* because (1) each method may have different advantages depending on the specific task and specific properties of the data set being analyzed, (2) the underlying technology is continually evolving and (3) noise levels in measurements do not always allow for fine discrimination between methods. Although various clustering methods can usefully organize tables of gene expression measurements, the resulting ordered but still massive collection of numbers remains difficult to assimilate. Hence clustering methods are combined with a graphical representation of the primary data by representing each data point with a color that quantitatively and qualitatively represents the original experimental observations. The end product is a representation of complex gene expression data that, through statistical organization and graphical display allows biologists to assimilate and explore the data in a natural and intuitive manner.

One of the most widely used unsupervised clustering technique for microarray data is hierarchical clustering, due to its simplicity and the ease of interpreting results from the output. Broadly, hierarchical clustering is divided into two types, agglomerative and divisive. In agglomerative clustering, each object (gene in case of microarray data) is initially placed in a group of its own. The two "closest" groups are combined into a single group. This is repeated until a single cluster encompassing all the objects remains. In divisive clustering, all objects are initially placed into a single group. The two objects that are in the same group but are "farthest" away are used as seed points for two groups. All objects in this group are placed into the new group that has the closest seed. This procedure continues until a threshold distance is reached. To determine when two objects are "close" or "far" various distance measures can be defined. Various measures such as euclidean distance, Manhattan distance, correlation coefficient have been used for measuring the distance between objects. We are interested in correlation coefficient since a seminal paper [7] uses this metric to perform agglomerative hierarchical clustering on microarray data. Also, many biologists ([1, 9, 13, 14, 17, 20, 22]) seem to use the software *Cluster*, which has been developed in [7].

1.1 The Microarray clustering problem

Depending on the way the distance between the clusters are computed, agglomerative hierarchical clustering can be divided into six types single linkage, complete linkage, average linkage, median, centroid and Ward's method. The method used in [7] is average linkage hierarchical clustering. Data from the microarray experiments are organized in the form of a matrix where, the rows correspond to the genes of interest and the columns correspond to the various experimental conditions or time points under which the expression levels are measured. The expression levels are stored as data in the matrix. Pearson correlation coefficient is used to measure the similarity between genes in [7]. If the number of genes to be clustered is N, the strategy used in [7] is to first calculate all possible pairs of similarities between genes. Each gene is initially thought to be in a cluster of its own. The closest pairs of genes are merged into a cluster and the similarity between the merged cluster and remaining clusters are computed. This step is repeated until a single cluster contains all the genes is left. It is easy to see that the run time of this naive algorithm is $O(N^3)$. There exists methods [8] which can be used to perform the clustering in $O(N^2)$ time, though the biology community seems to be unaware of them. As the microarray technology keeps improving the number of genes on a microarray increases and hence even $O(N^2)$ time can become unreasonable for large values of N. In this thesis, we show how to reduce the gene vectors to points in multi-dimensional space and perform the clustering in $O(Nlog\ N)$ time using the fairsplit tree data structure, and show that it is optimal. We also present a new serial algorithm, which takes $O(N^2)$ time to perform single linkage hierarchical clustering for points in space.

1.2 Organization of the thesis

The rest of the thesis is organized as follows: In Chapter 2, we describe the current methods of hierarchical clustering and review of relevant literature. Chapter 3 presents our main algorithmic ideas to perform the clustering in optimal time and Chapter 4 contains the new algorithm for single link hierarchical clustering. Conclusion and future work are presented in Chapter 5.

CHAPTER 2. LITERATURE REVIEW AND EXISTING APPROACHES

As previously mentioned there are two types of hierarchical clustering, agglomerative and divisive. Henceforth, we will be interested only in agglomerative hierarchical clustering since this is the most widely used technique for microarray data analysis. The reason being divisive clustering is computationally expensive when it comes to making decisions in dividing a cluster in two, given all possible choices. From now on, whenever we say hierarchical clustering, we imply agglomerative hierarchical clustering.

2.1 Types of hierarchical clustering

Clustering is done on a set of objects with associated description vectors, i.e points in multidimensional space. A dissimilarity or distance measure (d) is defined on these: $d(i, j) \geq 0$ and $d(i, j) = d(j, i)$ for objects or clusters i and j. Hierarchical clustering can be described informally as :

1. Determine all inter-object dissimilarities.
2. Form a cluster from two closest objects or clusters.
3. Recompute dissimilarities between new cluster and other objects or clusters.
4. Return to Step 2 until all objects are in one cluster.

Suppose there are N items to be clustered. Assuming d can be computed in constant time steps 1 and 2 requires $O(N^2)$ time. Step 3 can be completed in $O(N)$ time using Lance-Williams combinatorial formula. If the objects or clusters just merged are denoted by i and j and k is any other object or cluster, then the formula is

$$d(i+j,k) = a(i)d(i,k) + a(j)d(j,k) + b \times d(i,j) + c \times |d(i,k) - d(j,k)| \tag{2.1}$$

where values of a,b and c depend on the clustering strategy used as shown in the Table 2.1.

Table 2.1 Types of hierarchical clustering

Method	dissimilarity update formula	co-ordinates of center of cluster	Dissimilarity between cluster centers g_i and g_j
Single Link(nearest neighbor)	$a(i)$=0.5, b=0, c=-0.5	-	-
Complete Link(diameter)	$a(i)$=0.5, b=0, c=0.5	-	-
Average Link(group average)	$a(i) = \frac{\lvert i\rvert}{\lvert i+j\rvert}$	-	-
Median	$a(i)$=0.5, b=-0.25, c=0	$g = \frac{(g_i+g_j)}{2}$	$\lvert\lvert g_i - g_j\rvert\rvert^2$
Centroid	$a(i) = \frac{\lvert i\rvert}{(\lvert i\rvert+\lvert j\rvert)}$, $b = \frac{-\lvert i\rvert\lvert j\rvert}{(\lvert i\rvert+\lvert j\rvert)^2}$, c=0	$g = (\frac{\lvert i\rvert g_i+\lvert j\rvert g_j)}{(\lvert i\rvert+\lvert j\rvert)}$	$\lvert\lvert g_i - g_j\rvert\rvert^2$
Ward's Method	$a(i) = \frac{(\lvert i\rvert+\lvert k\rvert)}{(\lvert i\rvert+\lvert j\rvert+\lvert k\rvert)}$, $b = \frac{-\lvert k\rvert}{\lvert i\rvert+\lvert j\rvert+\lvert k\rvert}$, c=0	$g = \frac{(\lvert i\rvert g_i+\lvert j\rvert g_j)}{(\lvert i\rvert+\lvert j\rvert)}$	$\frac{\lvert i\rvert\lvert j\rvert}{(\lvert i\rvert+\lvert j\rvert)}\lvert\lvert g_i - g_j\rvert\rvert^2$

Since in each iteration the number of objects or clusters is reduced by one, it is easy to see that the above procedure takes $O(N^3)$ time. Depending on whether a cluster is represented by a *center* point or by a subgraph of interconnected points, the clustering algorithms can be divided into graph methods (single, complete and average link methods) and geometric methods (median, centroid and ward's method). It has been shown in [16] that for all clustering methods except centroid and median methods, clustering can be preformed in $O(N^2)$ time. The basic idea is the following. Let the nearest neighbor graph (NN-graph) be defined on a set of points p, whose directed edges (p,$NN(p)$) are such that $NN(p)$ is the nearest neighbor of p. If for points p and q, q=$NN(p)$ and p=$NN(q)$, then they are known as reciprocal nearest neighbors ($RNN's$). If i and j are any two $RNN's$, suppose we have a ρ such that for any other point k :

Algorithm 1 *Single Cluster Algorithm*

```
While only one point remains
    Start with arbitrary $i$, determine $j$=$NN(i)$, $k$=$NN(j)$,... until $q$=$NN(p)$,$p$=$NN(q)$. (also known as the NN-chain)
    Agglomerate RNNs $p$ and $q$, replacing with cluster center.
    Continue the NN-chain from the point in the chain prior to $p$ (or if the first two points in the NN-chain gave a pair of RNNs start a NN-chain from any point).
```

Figure 2.1 Single cluster algorithm for Ward's method

$$d(i,j) < \rho$$
$$d(i,k) > \rho$$
$$d(i+j,k) > \rho$$

Then, for all clustering methods except centroid and median methods it is easy to show that $d(i+j,k) > \rho$, using Lance-Williams recurrences. This is known as the *reducibility property*. The algorithm for the Ward's method can be given as in Figure 2.1

Using amortized analysis it can be shown that the above algorithm takes $O(N^2)$ time and the space required is $O(N)$ for geometric methods. Since there is no cluster center for the graph methods the algorithm can be generalized by storing the entire distance matrix, thus increasing the space to $O(N^2)$. The same result has been extended to all forms of hierarchical clustering (both geometric and graph methods) in [8] using a Quadtree like data structure (QTL), which is used for storing the nearest neighbors of points through insertion and deletion of points. QTL can be described in the following way: Group the points arbitrarily into pairs. Define the distance between two pairs to be the minimum of the four distances between objects in the pairs. Then, these pairs define a closest point problem with half as many points, which can be solved recursively. Each insertion or deletion causes $O(N)$ changes to the distance matrix of the points and leads to a single update in the recursive problem. Thus, we can maintain the closest pair among a set of N objects in time $O(N)$ per insertion and deletion, and $O(N^2)$ space. Hierarchical clustering can now be done as in Figure 2.2.

Algorithm 2 *Hierarchical Clustering Algorithm*

```
Create the QTL data structure for storing the closest pair among $N$ objects.
Repeat until a single cluster remains
    Merge the objects in the closest pair into one cluster.
    Insert this cluster into the data structure.
```

Figure 2.2 Generic hierarchical clustering using QTL data structure

Since there are $N-1$ merges of the closest pairs (clusters), it is easy to see that hierarchical clustering can be done in $O(N^2)$ time.

2.2 Existing approaches

A natural basis for organizing gene expression data is to group together genes with similar patterns of expression. The first step would be to adopt a mathematical description of similarity. A number of measures of similarity can be used such as Euclidean distance, angle, or dot products of the two D-dimensional vectors representing a series of D measurements. Eisen et al. [7] have used the standard correlation coefficient (i.e, the dot product of the two normalized vectors). This is because it conforms well to the intuitive biological notion of what it means for two genes to be co-expressed. The statistic captures *similarity* in shape but places no emphasis on the magnitude of the two series of measurements. Average linkage hierarchical clustering is used in [7] for clustering the genes. Relationship between objects (genes) are represented by a tree whose branch lengths reflect the degree of similarity between objects, as assessed by a pairwise similarity function described above. In sequence comparison these methods are used to infer the evolutionary history of the sequences being compared. No such underlying tree exists for patterns of genes. Such patterns are useful in their ability to represent varying degrees of similarity and more distant relationships between groups of closely related genes. The computed tree can be used to order the genes so that genes or groups of genes with similar expression patterns are adjacent. This binary tree is also called *dendrogram.* For example, if four genes a, b, c and d are clustered such that genes a and b are clustered first, followed by c and d, followed by the resulting two clusters being merged. The dendrogram in such a scenario can be visualized as Figure 2.3. The internal node x corresponds to the genes a and b being clustered, while the internal node y corresponds to the genes c and d being clustered together. The node z corresponds to the two clusters $\{a, b\}$ and $\{c, d\}$ being merged into a single resultant cluster.

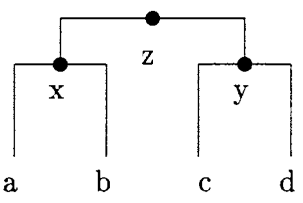

Figure 2.3 A simple dendrogram for four genes a,b,c and d.

Microarray data can be captured as a 2-Dimensional gene expression array, where the rows are genes which are used in the microarray experiment and the columns correspond to individual array experiments (e.g single time points or conditions), and each cell represents the measured Cy5/Cy3 fluorescence ratio at the corresponding target element on the array. All ratio values are log transformed (base 2 for simplicity) to treat inductions or repressions of identical magnitude as numerically equal but with opposite sign. Individual observations are rejected in some cases, on the basis of measurement quality parameters of the image analysis parameters. The gene similarity metric used is a form of correlation coefficient. Let G_i be the log-transformed data for gene G in condition i. For any two genes X and Y observed over a series of D conditions, a similarity score can be computed as :

$$S(X,Y) = \frac{1}{D}\sum_{i=1}^{D}(\frac{X_i - X_{offset}}{\Phi_Y})(\frac{Y_i - Y_{offset}}{\Phi_Y}) \tag{2.2}$$

where

$$\Phi_G = \sqrt{\sum_{i=1}^{D}\frac{(G_i - G_{offset})^2}{D}} \tag{2.3}$$

When G_{offset} is set to the mean of observations on G, then Φ_G becomes the standard deviation of G, and $S(X, Y)$ is exactly equal to the Pearson correlation coefficient of the observations of X and Y. Values of G_{offset} which are not the average over observations on G are used when there is an assumed unchanged or reference state represented by the value of G_{offset}, against which changes are to be analyzed. In [7], G_{offset} is set to 0, corresponding to a fluorescence ratio of 1.0. For any set of N genes, an upper-diagonal similarity matrix is computed by using the metric described previously, which contains the similarity score for all pairs of genes. The matrix is scanned to identify the highest value (representing the most similar pairs of genes). A node is created joining these two genes, and a gene expression profile is computed for the node by averaging observation for the joined elements (missing values are omitted and the two joined elements are weighted by the number of genes they contain). The similarity matrix is updated with this new node replacing the two joined elements, and the process is repeated $N - 1$ times until only a single element remains. Finding the maximum in matrix takes $O(N^2)$ time and creating the expression profile for the clustered node takes $O(N)$ time. Since these steps are repeated $N - 1$ times, the run time of the algorithm is $O(N^3)$.

CHAPTER 3. OPTIMAL HIERARCHICAL CLUSTERING

3.1 Terminology

Let D denote the number of experiments and N denote the number of genes $G_1, G_2, \ldots, G_N$ respectively. The gene expression matrix is an $N \times D$ matrix whose element g_{ik} denotes the expression level of gene G_i under experiment number k. Several metrics have been used to measure the similarity between two genes. With a *Euclidean metric*, one simply measures the Euclidean distance between the expression levels of two genes across a common set of experiments. But this measure does not appropriately capture the notion of co-regulated genes. For example, if the expression levels of two genes differ just by a multiplicative factor, they are considered co-regulated. However, the Euclidean distance between the two is proportional to this factor, contrary to the intuition of similarity.

On the other hand, the similarity metric used in [7] is based on the following correlation coefficient between two genes, G_i and G_j:

$$S(G_i, G_j) = \frac{1}{D}\sum_{k=1}^{D} \frac{(g_{ik} - m_i)}{\phi(G_i')} \frac{(g_{jk} - m_j)}{\phi(G_j')} = \frac{1}{D}\sum_{k=1}^{D} \frac{g'_{ik}}{\phi(G_i')} \frac{g'_{jk}}{\phi(G_j')} \tag{3.1}$$

where

$$\phi(G_i') = \sqrt{\frac{\sum_{k=1}^{D}(g_{ik} - m_i)^2}{D}} = \sqrt{\frac{\sum_{k=1}^{D} {g'_{ik}}^2}{D}} \tag{3.2}$$

and m_i reflects the reference value of gene G_i against which its changes are measured. The reference value modifies the original gene expression level g_{ik} to its displaced value g'_{ik}. If the reference value for each gene is set to the average of observations on that gene, then $\phi(G_i')$

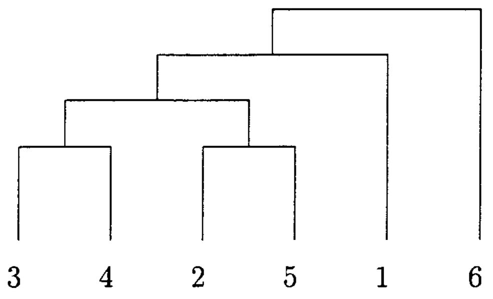

Figure 3.1 An example dendrogram.

is precisely the standard deviation and the above correlation function is called the *Pearson correlation coefficient.* According to this correlation based definition, $S(G_i, G_j) \in [0,1]\ \forall\ i, j$. In the case of the above example where one gene is simply proportional to another, it yields a coefficient equal to 1.

In the hierarchical algorithm adopted in [7], given N genes to be clustered, each gene G_k is initially considered to be in a cluster containing only that gene. It uses the above metric to compute the similarity score of the $O(N^2)$ possible pairs of genes and then includes a pair, say G_i and G_j, with the maximum score into a cluster, say G_{ij}. The rows corresponding to genes G_i and G_j are deleted and a row corresponding to the cluster G_{ij} is inserted into the gene expression matrix. This procedure is repeated until only one cluster remains. The elements of the row corresponding to a cluster G_{ij} are computed by taking the weighted average of the elements of the rows corresponding to the clusters i and j. The weights attached are the number of genes that are contained in the two joined clusters. The run time of this algorithm is clearly $O(N^3)$ since $O(N^2)$ comparisons are made in each of the $N-1$ iterations.

Clustering results are pictorially represented by what are called *dendrograms.* An example with six genes $\{G_1, G_2, \cdots, G_6\}$ is shown in Fig 3.1. The figure depicts the following sequence of clustering: $(G_3, G_4) \rightarrow (G_2, G_5) \rightarrow (G_{34}, G_{25}) \rightarrow (G_{2345}, G_1) \rightarrow (G_{12345}, G_6)$ where $G_{ij\cdots m}$ denotes a cluster that contains the genes $G_i, G_j, \cdots, G_m$.

3.2 Geometrical reduction

We treat the expression levels of a gene under the D experimental conditions, i.e., a row in the gene expression matrix, as a D-dimensional vector. The following definition of the dot product of two D-dimensional vectors, u and v, will be used:

$$u \cdot v = ||u|| \cdot ||v|| \ \cos\theta_{uv} \tag{3.3}$$

where $||w||^2 = \sum w_i^2$ denotes the norm of a vector w and $\cos\theta_{uv}$ is the angle between the vectors u and v and can be obtained by using the generalized law of cosines :

$$d^2(u,v) = ||u||^2 + ||v||^2 - 2\ ||u|| \cdot ||v|| \ \cos\theta_{uv} \tag{3.4}$$

where $d^2(u,v)$ is the L_2-distance between the tips of the vectors u and v drawn from the same origin. We will also use the notation $\widehat{V}$ for a normalized vector. Inserting Eqn (3.2) in Eqn (3.1) yields :

$$\begin{aligned} S(G_i,G_j) &= \sum_{k=1}^{D} \frac{(g_{ik}-m_i)}{\sqrt{\sum_{k=1}^{D}(g_{ik}-m_i)^2}} \frac{(g_{jk}-m_j)}{\sqrt{\sum_{k=1}^{D}(g_{jk}-m_j)^2}} \\ &= \widehat{G'}_i \cdot \widehat{G'}_j \\ &= ||\widehat{G'}_i|| \cdot ||\widehat{G'}_j|| \ \cos\theta_{\widehat{G'}_i\widehat{G'}_j} \\ &= \cos\theta_{\widehat{G'}_i\widehat{G'}_j} \end{aligned} \tag{3.5}$$

where we have used Eqn (3.3). Thus, the correlation coefficient between any two genes, G_i and G_j, can be interpreted as the cosine of the angle between the two D-dimensional normalized and possibly displaced gene vectors represented by $\widehat{G'}_i$ and $\widehat{G'}_j$. The above equation immediately implies that the correlation coefficient $S(G_i,G_j)$ does not depend on the length of the displaced gene vectors but *only* on the angle between them. In the following, we will set all the reference values to zero in the spirit of [7] in which case $\widehat{g'}_i \rightarrow \widehat{g}_i \ \forall\ i \in [1,N]$. It should be pointed out that all the subsequent arguments follow for non-zero reference values as well since there is a

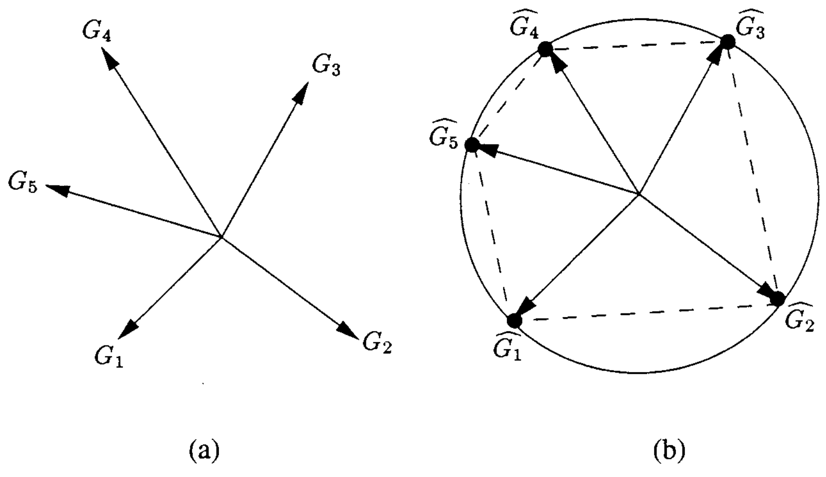

Figure 3.2 A 2D example where the radius of the circle is unity.

one-to-one mapping between a gene vector and its displaced counterpart.

Using Eqn (3.4) and Eqn (3.5), one can now find the distance $d(\widehat{G}_i, \widehat{G}_j)$ between the tips of the normalized gene vectors, $\widehat{G_i}$ and $\widehat{G_j}$ which yields :

$$d(\widehat{G}_i, \widehat{G}_j) = \sqrt{2(1 - \cos\theta_{\widehat{G}_i\widehat{G}_j})} = \sqrt{2(1 - S(G_i, G_j))} \tag{3.6}$$

Eqn (3.6) implies that a pair G_i, G_j maximizes the score $S(G_i, G_j)$ if and only if the pair $\widehat{G_i}, \widehat{G_j}$ minimizes the distance $d(\widehat{G_i}, \widehat{G_j})$. *Thus, the problem of finding a pair of D-dimensional gene vectors with maximum correlation from $\Theta(N^2)$ different pairs has been reduced to finding a closest pair among N points that lie on the surface of a D-dimensional hypersphere.* This is the well known dynamic closest pair problem.

A 2-dimensional example is shown in Fig 3.2. The gene vectors G_1, G_2, G_3, G_4 and G_5 in Fig 3.2(a) are unnormalized. The correlation coefficients between any pair of vectors in Fig 3.2(a) correspond to the cosine of the angle between them. Finding the smallest such angle corresponds to finding the closest pair of points on the circle as shown in Fig 3.2(b). The dotted lines in Fig 3.2(b) denote the Euclidean distances of Eqn (3.6).

3.3 An $O(N \log N)$ clustering algorithm

In this section, we will show how the above geometric interpretation immediately suggests an algorithm that reduces the $O(N^3)$ run time to $O(N \log N)$ run time without in any way disturbing the biological meaning of the procedure, described briefly in the next paragraph, adopted in [7].

To cluster a set $\mathcal{G}$ of N gene expression vectors $G_1, G_2, \ldots, G_N$, each vector is initially considered to be in a cluster with itself as the only member of that cluster. Thus, at the beginning, each of the genes can be viewed as a set of N singleton clusters, each represented by the only gene expression vector that it contains. Of these N clusters, a pair represented by, say G_i and G_j, that yields the highest correlation coefficient, $S(G_i, G_j)$ from the $\binom{N}{2}$ possible values are chosen to belong to a cluster that is represented by another gene expression vector G_{ij}. The components of the gene expression vector G_{ij} that represents this newly formed cluster are determined by taking a weighted average of the components, namely, G_i and G_j, respectively. The weight attributed to each of the merged clusters is the number of genes contained in the cluster divided by the sum of the total membership of the two joined clusters. Once this representative gene expression vector, G_{ij} is calculated, the clusters represented by G_i and G_j that took part in the formation of the most recent cluster G_{ij} are deleted and the vector G_{ij} is inserted into $\mathcal{G}$ thus reducing the cardinality of $\mathcal{G}$ by one. The above procedure is repeated and at every stage the cardinality of $\mathcal{G}$ decreases by one. The algorithm stops when $|\mathcal{G}| = 1$. Thus, there are $N - 1$ stages in the above clustering algorithm with each stage taking $O(|G_i|^2)$ run time where G_i denotes the set of clusters at the i^{th} stage. Since $|G_i| = N - i$, $i \in [0, N-1]$, the total run time is easily seen to be $O(N^3)$.

In this section, we will show how the above geometric interpretation immediately suggests an algorithm that reduces the $O(N^3)$ run time to $O(N \log N)$ run time without in any way disturbing the biological meaning of the procedure, described briefly in the next paragraph, adopted in [7].

If one can reduce the $O(|G_i|^2)$ run time to $O(\log |G_i|)$ at every clustering stage i, it immediately follows that the above clustering procedure can be accomplished in $O(N \log N)$ time.

Algorithm 3 *CLUSTER*($\mathcal{G}$)

```
Normalize each vector in $\mathcal{G}$.
$N = |\mathcal{G}|$
Build fair split tree on the $N$ points whose coordinates are components of the corresponding
normalized vectors.
while ($N > 1$)
    Find closest pair, say, points $i$ and $j$
    $i + j$ = merge $(i, j)$
    Delete $i$ and $j$ from $\mathcal{G}$
    Insert point $i + j$ into $\mathcal{G}$
    $N = N - 1$
```

Figure 3.3 An $O(N \log N)$ clustering algorithm.

The geometrical reduction described in aforementioned section achieves just that.

We propose the algorithm shown in Fig. 3.3 which requires three basic operations, namely:

- Given a set $\mathcal{G}$ of N points, find a closest pair.
- Delete two points from $\mathcal{G}$.
- Insert one point into $\mathcal{G}$.

Data structures that support each of the above operations in $O(\log |\mathcal{G}|)$ time already exist. We will use a data structure called *modified fair split tree* for our purpose, though any other spatial data structure that supports the above operations in $O(\log |\mathcal{G}|)$ time will suffice. For further details about the above data structure and the operations defined on it, we refer the reader to the original sources [2, 3]. It is shown in [2] that a modified fair split tree can be built on N points in $O(N \log N)$ time. In addition, this data structure supports deletion and insertion operations in $O(\log N)$ time, respectively. Finding the closest pair using the modified fair split tree takes $O(\log N)$ time as well. All the aforementioned run times have constant factors dependent on the number of dimensions, D. Our algorithm for the clustering of N gene expression vectors is summarized in Fig. 3.3. The input to the algorithm is the set $\mathcal{G}$ of all the D-dimensional gene vectors. Its run time is easily seen to be $O(N \log N)$. The space requirement is $O(N)$ as shown in [2].

```
// Create a list in each leaf $i$ with $x_i$ for that leaf as the only element
Algorithm 4 SORT($v$)
  if ($v$ is a leaf)
      return list in $v$
  else
      left_list=SORT(left_child($v$))
      right_list=SORT(right_child($v$))
      if (rightmost element in left_list < leftmost element in right_list)
          concatenate list in left_child($v$) and list in right_child($v$)
      else
          concatenate list in right_child($v$) and list in left_child($v$)
      return the concatenated list
```

Figure 3.4 Algorithm *SORT*(v). *SORT*(v) is initially called with the root.

3.4 Proof of optimality

Optimality of the algorithm can be shown using a reduction from sorting. The required $\Omega(N \log N)$ lower bound can be established even for the 2D case. Consider a set of N distinct real numbers between 0 and 1. Construct an $N \times 2$ gene expression matrix in which the i^{th} row is (x_i, y_i) where $y_i = \sqrt{1 - x_i^2}$ in $O(N)$ time. Applying a clustering algorithm on this gene expression matrix will yield a dendrogram T which is a full binary tree with root r and N leaves. Each leaf in T represents a two dimensional gene vector $G_i \equiv (x_i, y_i)$. Now consider the sorting algorithm in Fig 3.4. It is easy to see that since the size of T is $O(N)$, *SORT*(v) runs in linear time. Correctness of *SORT*(v) follows from the facts that: **(a)** if G_i and G_j are clustered, then x_i and x_j are neighbors in the final sorted order. To prove this, note that if G_i and G_j are clustered by the algorithm, then it follows from the correctness of the clustering algorithm that $S(G_i, G_j) > S(G_i, G_k) \Rightarrow d(G_i, G_j) < d(G_i, G_k)\ \forall\ k \neq i, j$. Without loss of generality, assume that $x_i < x_j$. Let us assume that $\exists\ G_k \equiv (x_k, y_k)$ such that $x_i < x_k < x_j$. Since $y_i = \sqrt{1 - x_i^2}\ \forall\ 1 \leq i \leq N$, it follows that $y_j < y_k < y_i$. Therefore, $d(G_i, G_k) = (x_k - x_i)^2 + (y_k - y_i)^2 < (x_j - x_i)^2 + (y_k - y_i)^2 < (x_j - x_i)^2 + (y_j - y_i)^2 = d(G_i, G_j)$. But $d(G_i, G_k) < d(G_i, G_j) \Rightarrow S(G_i, G_j) < S(G_i, G_k)$ which contradicts the fact that G_i and G_j were clustered by the algorithm. **(b)** at each internal node of T, concatenation of two sorted lists conserves the order in the resulting list. The clustering algorithm that yielded the dendrogram T must then be $\Omega(N \log N)$ in order not to violate the algebraic decision tree model for the lower bound of the problem of comparison based sorting, thus, proving the optimality of $CLUSTER(\mathcal{G})$. It should be noted that we proved a lower bound by restricting attention to the 2D case since the lower bound for the restricted case also serves as a lower bound for the general problem and the lower bound for the restricted case matches the upper bound as given by $CLUSTER(\mathcal{G})$ for any number of dimensions.

CHAPTER 4. SINGLE LINKAGE HIERARCHICAL CLUSTERING

4.1 Overview

The single link method is the oldest hierarchical clustering method and one of the most widely used hierarchical methods because of computationally efficient algorithms. It is also of great interest for point pattern recognition. A wide range of algorithms have, in fact, been developed for single linkage method [19]. Algorithms reviewed in [19] have complexities ranging from $O(Nlog\ N)$ to $O(N^5)$. Many of these algorithms have first constructed the *Minimum Spanning Tree* (*MST*), and subsequently transformed this into the single link hierarchy. The two problems of single linkage clustering and the *MST* are closely related. Information is lost in transforming the MST into the hierarchy, so that the reverse transformation is not possible. An efficient $O(N^2)$ worst case algorithm for transforming the *MST* into the hierarchy is described in [18]. A number of authors [11] found the mathematical properties of this method so appealing that they preferred it to all other hierarchical methods. The properties which are not shared by other methods as pointed in [11] are

1. Every partition has classes which are optimal with reference to the connectivity criterion used. Partitions obtained from the minimum variance method, in contrast are suboptimal with respect to intra cluster minimum variance.

2. Monotonic transformation of input dissimilarities (i.e a transformation which preserves order) has no effect on the hierarchy. This may be of importance where a question is raised over the scaling of directly-constructed dissimilarities.

3. Small changes in the input dissimilarities produce small changes in the hierarchy produced. Thus, the single link method is a stable method.

4. It is easy to graphically represent the results of linkage based clustering methods on 2-dimensional data and the hierarchical method or the minimal spanning tree are very suitable for many types of pattern recognition problems.

The automatic recognition of groups of points is an important problem in the pattern recognition area. For example, in Figure 4.1 all hierarchical clustering algorithms perform well when presented with well-separated, compact clusters. For elongated clusters , the single link method with its chaining effect would be ideally suited. For linking and touching globular groups, single link may not be of direct use. The minimum variance method should however find the clusters. Finally, in case of concentric groups or of groups characterized by differing densities, the *MST* may be used. In the former case, a large link will indicate whether the *MST* must be broken to form two components. In the latter case, a histogram of edge dissimilarity weights should uncover two distinguishable sets of dissimilarities: edges of small dissimilarity will relate to the high density part of the point pattern, and edges of greater dissimilarity will relate to the low density region. Deleting all edges of dissimilarity weight greater than some threshold in the *MST* causes the resultant tree to connect only high density points.

4.2 Existing algorithms

Since the problem of performing the single link clustering is equivalent to finding the *MST* of the given points, most of the algorithms first find the *MST* of the input points. The number of points N correspond to the vertices V of the graph. A dissimilarity measure is used to find the edge weights between the vertices. The set of edges between all possible pairs of vertices is denoted by E. The problem of finding a minimum spanning tree of a given weighted graph has been studied extensively and numerous (sequential and randomized) algorithms have been devised. Asymptotically optimal randomized algorithms have been developed [12], which run in time $O(|V| + |E|)$ with high probability. A deterministic algorithm with a run time of $O(|E|\alpha(|E|, |V|))$ has been given in [4] where α is the inverse Ackerman's function. Shamos and Hoey [21] have presented an $O(Nlog\ N)$ time algorithm for this problem assuming that the points are from a two-dimensional space. Yao [27] considers the euclidean *MST*

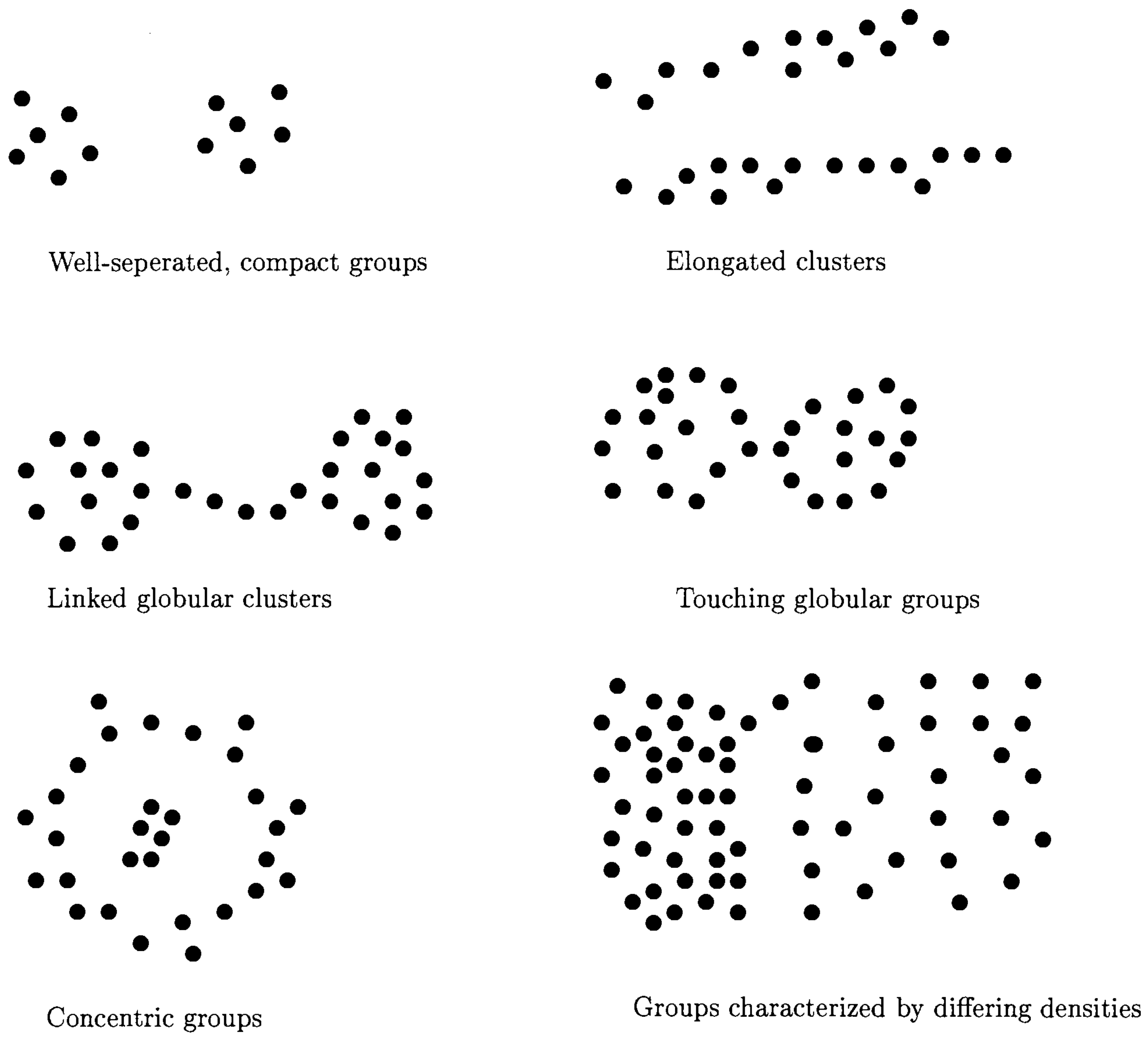

Figure 4.1 Point patterns whose constituent groups are to be automatically recognized.

Algorithm 5 *Single Link Clustering*

```
Compute the array $D(i,j)$ which contains the (dis)similarity between points $i$ and $j$,
for all possible pairs $(i,j)$.
Compute nearest neighbor $N(i)$ for every point $i$.
Repeat $N-1$ times
    Determine $i$,$j$ such that $D(i,j)$ is minimized.
    Agglomerate clusters $i$ and $j$.
    Update each $D(i,j)$ and $N(i)$ as necessary.
```

Figure 4.2 A simple $O(N^2)$ single link clustering algorithm.

problem in d dimensions ($d \geq 3$). In particular, he presented an algorithm with a run time of $O(N^{2-a(d)}(log\ N)^{1-a(d)})$ where $a(d) = 2^{-(d+1)}$. When $d = 3$ an improved algorithm with a run time of $O(Nlog\ N)^{1.8}$ has also been given in [27]. Most of these algorithms run in almost $O(N^2)$ time and are impractical for high dimensional data. A much simpler algorithm which can be employed for single linkage clustering is shown in Figure 4.2.

Computing the arrays $D(i,j)$ and $N(i)$ takes $O(N^2)$ time. Using these arrays computing two closest clusters can be performed in $O(N)$ time by examining each cluster's nearest neighbor. To agglomerate clusters, we can simply store which clusters were agglomerated and update the arrays. The single link metric has the property that if we agglomerate clusters i and j into $i+j$, any cluster that had either i or j as its nearest neighbor now has $i+j$ as its nearest neighbor. This property allows us to update the arrays in $O(N)$ time for the single link metric, yielding an $O(N^2)$ time algorithm.

4.3 New algorithm

In this section, we propose a new algorithm for performing single link hierarchical clustering.

Algorithm 6 *A New Algorithm*

```
Find the nearest neighbor $N(i)$ for every point $i$.
Sort these nearest neighbor distances.
Make a cut in this sorted list.
Repeat until a single cluster remains or cut covers all the points.
    Perform the clustering for points on the left side of the cut and build the dendrogram.
    Cut the dendrogram according to a specific criterion(which will be defined shortly).
    Cluster parts of the dendrogram, find nearest neighbors of these clusters and merge
    this list with the sorted list on the right side of the cut.
```

Figure 4.3 A new $O(N^2)$ single link clustering algorithm.

4.4 Criterion for cutting the dendrogram

A dendrogram of N leaves can be considered a binary tree with N leaves and N-1 internal nodes. Henceforth, the terms dendrogram and tree will be used interchangeably when the difference is clear from the context. The dendrogram is decomposed according to Figure 4.4.

Thus the dendrogram is decomposed into a set of connected components, which we claim are correct with respect to the final dendrogram of the input points. In other words, we stop cutting the dendrogram whenever we encounter an internal node which has atleast one child as a leaf.

Claim 1 *Cutting the dendrogram according to Figure 4.4 yields correct clusters according to the final dendrogram.*

Proof: Initially, since we are sorting the nearest neighbor distances, all the points to the left side of the cut have their nearest neighbors on the same side of the cut. We stop breaking the dendrogram when we encounter an internal node which has atleast one child as a leaf. Suppose at the point of stopping, the internal node say X has as its children a leaf node i and subtree rooted at Y (see Fig(4.5)). Suppose i was clustered with the set containing leaves underneath node Y because of its distance $d(i, j)$ with a leaf say j. We are claiming that the cluster under

Algorithm 7 *Algorithm to decompose the dendrogram*

```
Push the node onto the stack.
While stack is not empty
    Pop a node from the stack.
    If (node is a leaf)
        End and exit out of the loop.
    else
        Push the children of the node onto the stack.
        Delete the set of edges between the node and children of the node.
```

Figure 4.4 Algorithm is initially called with the root of the tree.

X is correct. Suppose for contradiction, cluster under X is not correct. There are two cases possible.

Case I: Cluster under node Y should have clustered with a different cluster before it clustered with i. In which case the pair involved in clustering Y with other cluster must have been to left of pair (i, j) in the sorted list. Hence, that cluster must have clustered with cluster under Y before it clustered with i, thus contradicting the clustering process which is applied to all the points on the left side of the cut.

Case II: Node i must be clustered with a different cluster. This case can be proved in a similar way as the first case. ■

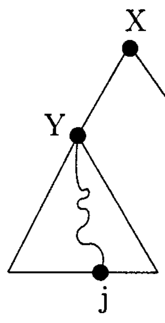

Figure 4.5 An example binary tree

4.5 Run time analysis

Finding the nearest neighbor of every point takes $O(N^2)$ time, if there are N points to start with. Sorting these N distances takes $O(Nlog\ N)$ times using any of the common comparison based sorting algorithms like merge-sort or heap-sort. Suppose the *cut* is made such that αN points are on the left side of the *cut* and $(1-\alpha)N$ points are on the right side of the *cut*. The number of points on the left side are reduced by atleast half, since every point is atleast clustered with its nearest neighbor. The recurrence can be given as:

$$T(N) = T(\alpha N) + T((1-\frac{\alpha}{2})N) + O(N^2) \tag{4.1}$$

To solve this recurrence, we will use induction. Let $T(N) \leq cN^2$ for some constant c and $O(N^2) = kN^2$ for some constant k. Substituting in the equation 4.1 we get

$$c(\alpha N)^2 + c((1-\frac{\alpha}{2})N)^2 + kN^2 \leq cN^2 \Rightarrow k \leq c(\alpha - \frac{5\alpha^2}{4}) \Rightarrow c \geq \frac{4k}{(4\alpha - 5\alpha^2)} \tag{4.2}$$

In other words, for any k, we can choose α such that the above inequality holds, thus proving the induction. Hence, the total run time of the algorithm is $O(N^2)$.

4.6 Optimal value of α

We would like to find an optimal value of α, which will decide where the *cut* must be made in the sorted list. According to equation 4.2, in order to minimize the value of c, $4\alpha - 5\alpha^2$ must be maximized. Differentiating with respect to α and equating to 0, we have

$$4 - 10\alpha = 0 \Leftrightarrow \alpha = \frac{2}{5} \tag{4.3}$$

Thus, the optimal point for making the cut in the sorted list of nearest neighbors is after $\lfloor \frac{2N}{5} \rfloor$ elements.

4.7 An example

In this section, we present an implementation of the algorithm on a small data set. Consider a data set of 10 points $a, b, ..., j$ with the distance matrix as shown in Figure 4.1.

Table 4.1 A sample data set

	a	b	c	d	e	f	g	h	i	j
a	∞	5	9	35	30	25	34	43	51	60
b	5	∞	12	36	31	29	35	42	50	61
c	9	12	∞	37	32	30	36	41	52	62
d	35	36	37	∞	8	20	12	40	53	63
e	30	31	32	8	∞	17	24	44	56	64
f	25	29	30	20	17	∞	10	45	54	65
g	34	35	36	12	24	10	∞	46	55	65
h	43	42	41	40	44	45	46	∞	60	67
i	51	50	52	53	56	54	55	60	∞	15
j	60	61	62	63	64	65	66	67	15	∞

According to the first step of the algorithm, the sorted nearest neighbor list is shown in Figure 4.6. The head of the arrow is the point and the tail points to the nearest neighbor of the point with the distance to the nearest neighbor stored along the arrow.

$$\begin{tikzcd}
a \arrow[d, "5"'] & b \arrow[d, "5"'] & d \arrow[d, "8"'] & e \arrow[d, "8"'] & c \arrow[d, "9"'] & f \arrow[d, "10"'] & g \arrow[d, "10"'] & i \arrow[d, "15"'] & j \arrow[d, "15"'] & h \arrow[d, "40"'] \\
b & a & e & d & a & g & f & j & i & d
\end{tikzcd}$$

Figure 4.6 Sorted list of nearest neighbors.

As proved earlier, to optimally cut this sorted list, we need to cut after $\frac{2}{5}$ times the length of the list. The dendrogram for the first four points in the list is shown in Figure 4.7

According to the criterion for cutting the dendrogram, starting from the root (Z in our case), we stop when encountered with a node containing atleast one leaf as a child. In this case we stop when after cutting the links between Z to X and Z to Y. We claim the the pairs

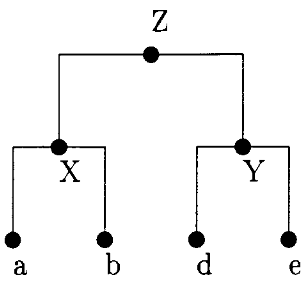

Figure 4.7 Dendrogram for points on left side of the cut.

(a, b) and (d, e) are correct with respect to the original dendrogram which is trivially true. Two clusters are formed by clustering a and b into one cluster and d and e into another cluster. This process is repeated until we build the final dendrogram which is shown in Figure 4.8.

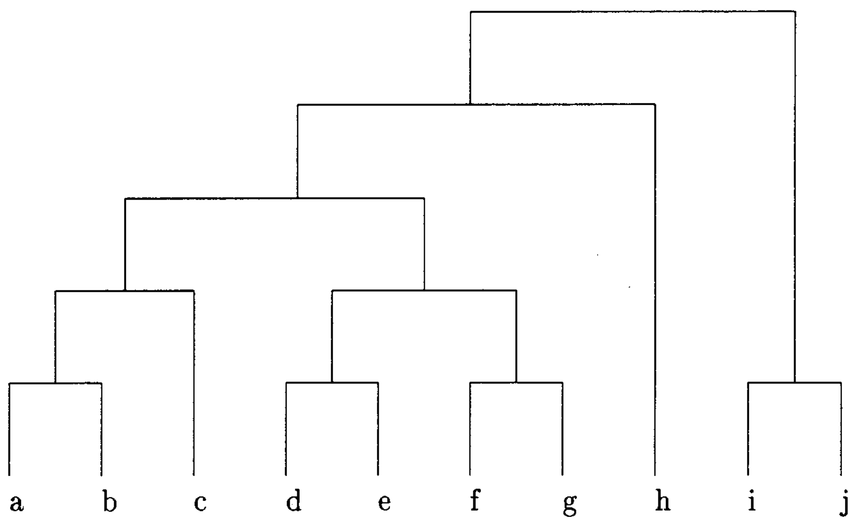

Figure 4.8 Final dendrogram.

CHAPTER 5. CONCLUSION AND FUTURE WORK

In this thesis, we have shown that the interpretation of the correlation coefficient of two row vectors of a gene expression matrix as the cosine of the angle between the two row vectors can be exploited to convert the problem of clustering genes based on a correlation coefficient metric into a dynamic closest pair problem. Such a reduction allows a $\Theta(N \log N)$ algorithm over the existing $O(N^3)$ algorithm. The geometric algorithms used for the dynamic closest pair problem have a run-time dependence on D that is exponential. This could be problematic if D is large. One possible approach is to use approximate closest pair algorithms which have a linear dependence on D. Note that the closest pair algorithm used in the paper is general in nature, and does not take into account the special structure of the microarray hierarchical clustering problem where all the points lie on the surface of a D-dimensional unit hypersphere. It is an interesting open problem to see if better data structures that exploit this specific geometry can be designed to realize a more graceful dependence on the number of dimensions. A new serial algorithm for single linkage hierarchical clustering has been developed. Since hierarchical clustering is widely used for clustering microarray data, parallel algorithms would be of great help to cluster large data sets quickly. Most of the parallel algorithms developed are for theoretical models. Work done by parallel algorithms which have been implemented on microarray data is $O(N^3)$, which is worse than $O(N^2)$ time taken by serial algorithms. It is an open problem if efficient and scalable parallel algorithms can be developed for these sequential algorithms.

BIBLIOGRAPHY

[1] A. A. Alizadeh, M. B. Eisen, R. E. Davis, and C. Ma *et al.* Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.

[2] S.N. Bespamyatnikh. An optimal algorithm for closest pair maintenance. *Discrete Computational Geometry*, 19:175–195, 1998.

[3] P. B. Callahan and S. R. Kosaraju. A decomposition of multi-dimensional point-sets with applications to k-nearest-neighbors and n-body potential fields. In *Proc. 24th Annual ACM Symposium on Theory of Computing*, pages 546–556, 1992.

[4] B. Chazelle. A minimum spanning tree algorithm with inverse-ackerman type complexity. *Journal of the ACM*, 47(6):1028–1047, 2000.

[5] S. Chu, J. DeRisi, Michael B. Eisen, and J.Mulholland*et al.* The transcriptional program of sporulation in budding yeast. *Science*, 282:699–705, 1998.

[6] D.D. Shoemaker and E.E. Schadt, C.D. Armour and Y.D. He *et al.* Experimental annotation of the human genome using microarray technology. *Nature*, 409:922–927, 2001.

[7] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. National Academy of Sciences*, 95(25):14863–14868, 1998.

[8] David Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 619–628, 1998.

[9] Audrey P. Gasch, Paul T. Spellman, Camilla M. Kao, and Orna Carmel-Harel *et al.* Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology Cell*, 11(12):4241–4257, 2000.

[10] V.R. Iyer, M.B. Eisen, D.T. Ross, and G. Schuler *et al.* The transcriptional program in the response of human fibroblast to serum. *Science*, 283:83–87, 1999.

[11] N. Jardine and R. Sibson. *Mathematical Taxonomy.* Wiley, NewYork, 1971.

[12] D.R. Karger, Philip N. Klein, and R.E. Tarjan. A randomized linear time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328, 1995.

[13] S. Kawasaki, C. Borchert, M. Deyholos, and H. Wong *et al.* Gene expression profiles during the initial phase of salt stress in rice. *Plant Cell*, 13(4):889–906, 2001.

[14] A. B. Khodursky, B. J. Peter, D. Botstein, and Patrick O. Brown *et al.* DNA microarray analysis of physiological conditions and genetic changes that affect tryptophan metabolism in *escherichia coli. Proc. of the National Academy of Sciences*, 97:12170–12175, 2000.

[15] T. Kohonen. *Self-organizing maps.* Springer-Verlag, Berlin, 1997.

[16] F. Murtagh. An Survey of Recent Advances in Hierarchical Clustering Algorithms. *The Computer Journal*, 26:354–359, 1983.

[17] Charles M. Perou, Stefanie S. Jeffrey, Matt van de rijn, and Christian A. Rees *et al.* Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. *Proc. National Academy of Sciences*, 96:9212–9217, 1999.

[18] F.J. Rohlf. Hierarchical clustering using the minimum spanning tree. *The Computer Journal*, 16:93–95, 1975.

[19] F.J Rohlf. *Single Link Clustering Algorithms*, volume 2. Handbook of Statistics, 1982.

[20] D.T Ross, U. Scherf, M.B. Eisen, and C.M. Perou *et al.* Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics*, 24:227–235, 2000.

[21] M.I Shamos and D.J. Hoey. Closest point problems. In *Proc. 16th Annual IEEE Symposium on Foundations of Computer Science*, pages 151–162, 1975.

[22] T. Sorlie, C.M. Perou, R. Tibshirani, and T. Aas *et al.* Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proc. National Academy of Sciences*, 98(19):10869–10874, 2001.

[23] Paul T. Spellman, G. Sherlock, M.Q. Zhang, and R.I Vishwanath *et al.* Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[24] P. Tamayo, D. Slonim, J. Mesirov, and Q. Zhu *et al.* Interpreting patterns of gene expression with self-organizing maps: Methods and an application to hematopoietic differentiation. *Proc. National Academy of Sciences*, 96:2907–2912, 1999.

[25] S. Tavazoie, J.D. Hughes, M.J. Campbell, and R.J. Cho *et al.* Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285, 1999.

[26] J. Craig Venter, Mark D. Adams, Eugene W. Myers, and Peter W. Li *et al.* The sequence of human genome. *Science*, 291(5507):1304–1351, 2001.

[27] A. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

ACKNOWLEDGEMENTS

I sincerely thank my major professor Dr.Srinivas Aluru, for his constant support and encouragement throughout this research. He has been a source of inspiration for me to achieve my research goals. Without his constant guidance this research would not have been possible. He showed me how to think analytically and the fun involved in doing research.

I would like to express my thanks to Sudip, who played a major part in this research. Hopefully, all the afternoons we spent cracking our heads to make some headway have been fruitful.

I would like to specially thank my labmate mahesh, who has been a mentor for me and helped me come out of the occasional bouts of frustration. His constant motivation helped me a lot.

I would like to thank ananth, for constantly helping me out with my doubts and taking time to discuss my research. His clear way of analyzing showed me how research should be done. Discussions with him always brought out new ideas and helped me gain more insight into the problems we discussed.

I would like to express my gratitude to all my labmates pang, scott and sarah.

I am indebted to my parents and brother for providing me with support and affection at all times. Also, huge love for my friends tejeswi, vasudha, swetha, ravi, sandeep, pavan, deepak and allada for their constant support and without whom my stay here at Iowa State University would have been dull and boring.