Analysis of Gene Expression Data **Spring Semester, 2004**

Lecture 5: April 15, 2004

Lecturer: Ron Shamir *Scribe: Michal Ozery-Flato and Israel Steinfeld*[1]

5.1 Introduction

5.1.1 Functional Genomics

Having reached the end of the Human Genome Project, the question that needs to be asked is: "What's next?". The complete sequencing of the Human Genome is an immense task, which is now nearing completion. While much work remains to be done even there, there are a number of areas this knowledge opens up to research, which have thus far been nearly impossible to pursue. Among those is "functional genomics" - the search for understanding the functionality of specific genes, their relations to diseases, their associated proteins and their participation in biological processes. Most of the knowledge gained so far in this area is the result of painstaking research of specific genes and proteins, based on complex biological experiments and homologies to known genes in other species. This "Reductionist" approach to functional genomics is hypothesis driven (i.e., it can be used to check an existing hypothesis, but not to suggest a new one). The advancements in both biological and computational techniques are now beginning to make possible a new approach: the "Holistic" research paradigm. This approach is based on high-throughput methods: global gene expression profiling ("transcriptome analysis") and wide-scale protein profiling ("proteome analysis"). In the holistic approach, a researcher simultaneously measures a very large number of gene expression levels throughout a biological process, thereby obtaining insight into the functions and correlations between genes on a global level. Unlike the reductionist approach, these methods can generate hypotheses themselves.

5.1.2 Representation of gene expression data

Gene expression data can be represented as a real matrix, called the *raw data matrix*. Each row in the matrix contains data regarding a specific gene, and each column represents a condition, or a tissues profile. Thus, R_{ij} is the expression level for gene i, at condition j. The expression data can represent ratios, absolute values, or distributions. The expression pattern of a gene i is the i^{th} row of R_{ij}. The expression pattern of a condition j is the j^{th} column of R_{ij}.

[1]Based in part on a scribe by Dror Fidler and Shahar Harrusi, April 2002, and on a scribe by Giora Sternberg and Ron Gabo, May 2002.

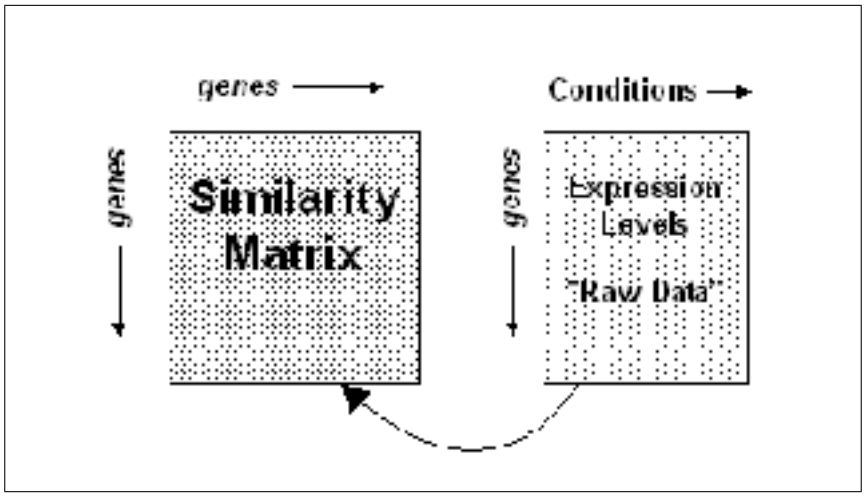

Figure 5.1: The raw data matrix maps conditions with gene expression. The *Similarity matrix* is derived from the raw data matrix, according to a similarity, or distance function.

In some clustering algorithms the raw data matrix is preprocessed to compute a *similarity* matrix S, where S_{ij} reflects the similarity of the expression patterns of gene i and gene j. Note that the similarity matrix is larger than the raw data matrix since there are usually much more genes than conditions. Figure 5.1 shows the data and similarity matrices.

5.1.3 Clustering applications

The analysis of expression profiles of genes or conditions can be helpful for many purposes, such as:

- Deducing functions of unknown genes from known genes with similar expression patterns.
- Identifying disease profiles - tissues with similar disease should yield similar expression patterns.
- Deciphering regulatory mechanisms - co-expression of genes implies co-regulation.
- and more ...

5.1.4 The Clustering Problem

Genes are said to be similar if their expression patterns resemble, and non-similar otherwise. The goal of gene clustering process is to partition the genes into distinct sets such that

genes that are assigned to a same cluster should be similar, while genes assigned to different clusters should be non-similar. Usually there is no one solution that is the "true"/correct mathematical solution for this problem, but a good clustering solution should have two merits:

1. *Homogeneity*: measures the similarity between genes assigned to the same cluster.
2. *Separation*: measures the distantnce/dis-similarity between the clusters. Each cluster should represent a unique expression pattern. If two clusters have similar expression patterns, then probably it would be better to unite them into one cluster.

Note that these two measures are in a way opposite- if you wish to increase the homogeneity of the clusters you would increase the number of clusters, but in the price of reducing the separation.

Clustering methods have been used in a vast number of fields. We can distinguish between two types of clustering methods:

Agglomerative These methods build the clusters by looking at small groups of elements and performing calculations on them in order to construct larger groups. The Hierarchal methods described here are of this sort.

Divisive A different approach which analyzes large groups of elements in order to divide the data into smaller groups and eventually reach the desired clusters. We shall see non hierarchal techniques which use this approach.

5.2 Hierarchic Clustering

This agglomerative approach attempts to place the input elements in a tree hierarchy structure in which the distance within the tree reflects element similarity. The elements are located at the leaves of the tree. Thus, the closer the elements are in the tree, the more similar they are.

Advantages of hierarchal methods :

1. A single coherent global picture.
2. Intuitive for biologists. (similar representation is used in Phylogeny).

Disadvantages of hierarchic methods :

1. There is no explicit partition into clusters.
2. A human biologist with extensive knowledge might find it impossible to make sense of the data just by looking at the tree, due to the size of the data, and the number of errors.

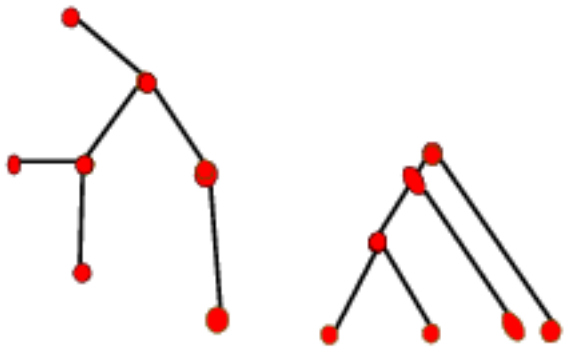

Figure 5.2: Hierarchal data can be represented as a rooted or un rooted tree

5.2.1 Hierarchical Representations

As was explained, a hierarchic representation uses a tree structure, in which the actual data is represented at the leaves. The tree can be rooted or not. A particular tree representation is a *dendrogram*. The algorithms for hierarchic clustering merge similar clusters, and compute the new distances for the merged cluster. Hence, if i is clustered with j, and both are not similar to r, then $D(i,r) = D(j,r)$ even though $D(i,j) > 0$. ($D(n,m)$ is the distance function) (See Figure 5.3).

5.2.2 Neighbor Joining Algorithm

A simple algorithm, based on neighbor merging, is due to Saitou and Nei [10]. The input matrix is the distance matrix between elements. Initially each element is a cluster. At each iteration we merge similar elements, and compute the new distances for the merged elements. When the algorithm has finished we represent the results as a tree in which similar elements are near.

The Neighbor Joining Algorithm :

1. Input : The Distance matrix D_{ij}.
2. Find elements r,s such that $D_{rs} = \min_{ij}(D_{ij})$.
3. Merge clusters r,s.

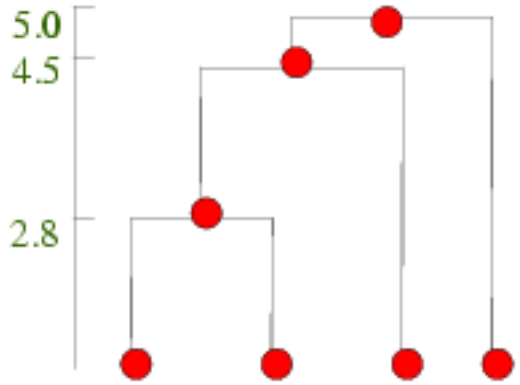

Figure 5.3: In a dendrogram, distances are represented on the y-axis. Denote the leaves a, b, c, d (from left to right). Then $D(a, b) = 2.8$, $D(a, c) = D(b, c) = 4.5$, $D(b, d) = D(c, d) = 5.0$

4. Delete elements r,s, and add a new element t with :

$$D_{it} = D_{ti} = \frac{D_{ir} + D_{is} - D_{rs}}{2}$$

5. Repeat, until one element is left.

6. Present the hierarchy as a tree with similar elements near each other.

5.2.3 Average linkage

Average linkage is a modification of the Neighbor Joining algorithm. The idea is the same but when computing the new distances of created clusters, the sizes of the clusters that are merged are taken into consideration. This algorithm was developed by Lance and Williams [7], and Sokal and Michener [13].

The Average linkage Algorithm :

1. Input : The Distance matrix D_{ij}, initial cluster sizes n_r.

2. Iteration k : The same as the Neighbor Joining algorithm with the exception that the distance from a new element t is defined by:

$$D_{it} = D_{ti} = \frac{n_r}{n_r + n_s} \cdot D_{ir} + \frac{n_s}{n_r + n_s} \cdot D_{is}$$

5.2.4 A General Framework

Lance and Williams, [7] also designed a general framework for hierarchal cluster merging algorithms. In their framework the distance calculating function is :

$$D_{it} = D_{ti} = \alpha_r D_{ir} + \alpha_s D_{is} + \gamma |D_{ir} - D_{is}|$$

In the Average Linkage algorithm :

$$\gamma = 0$$

$$\alpha_r = \frac{n_r}{n_r + n_s}$$

$$\alpha_s = \frac{n_s}{n_r + n_s}$$

5.2.5 Hierarchical clustering of gene expression data

A series of experiments were performed on real gene expression data, by Eisen et al. [4] The goal was to check the growth response of starved human fibroblast cells, which were given serum. 8600 gene levels were monitored over 13 time-points (about two cell cycles). The original data of test to reference ratios was first log transformed, and then normalized, to have mean 0 and variance 1. Or formally:
$t_{ij}-$ fluorescence levels of target gene i in condition j.
$r_{ij}-$ same for reference.
$D_{ij} = \log(\frac{t_{ij}}{r_{ij}})$.
$N_{ij} = \frac{[D_{ij} - E(D_i]}{std(D_i)}$. the normalized levels.

The similarity matrix was constructed from N_{ij} as follows :

$$S_{kl} = \frac{\sum_j N_{kj} \cdot N_{lj}}{Ncond}$$

Where *Ncond* is the number of conditions checked.
The Average Linkage method was applied on the similarity matrix, Where genes with low activity (70%) were not participating in the analysis. The tree is presented by ordering the leaves according to increasing subtree weight (see Figure 5.4). The weight can be average expression level, time of maximal induction, or any other. Figure 5.5 demonstrates the different output given by hierarchical when clustering gene expression data, and random data.

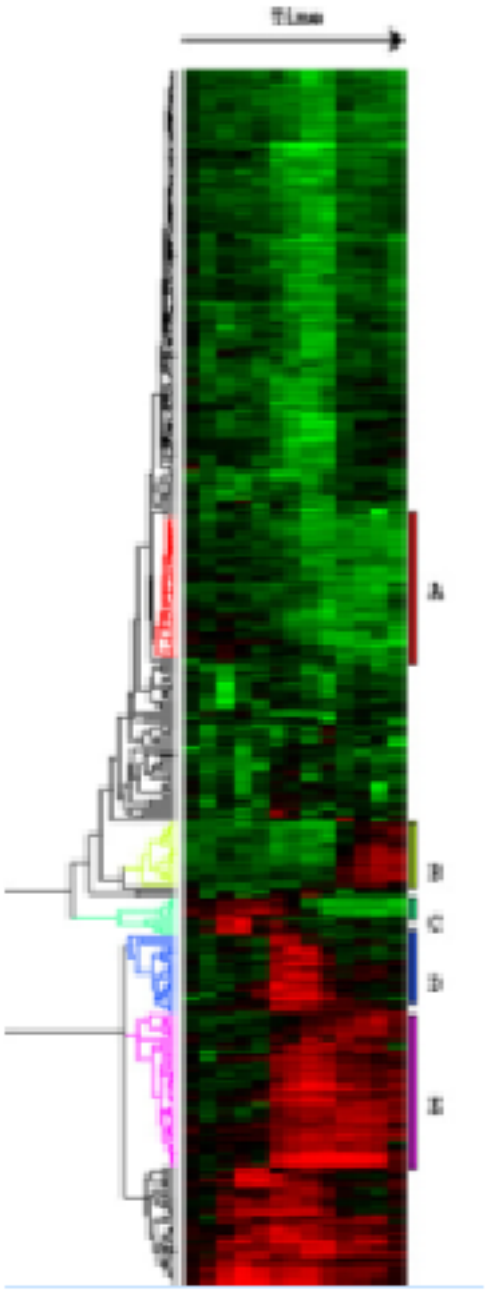

Figure 5.4: Source: [4]. The dendrogram resulting from the starved human fibroblast cells experiment. five major clusters can be seen, and many non clustered genes. The genes in the five groups serve similar functions : (A) cholesterol biosynthesis, (B) the cell cycle, (C) the immediate-early response, (D) signaling and angiogenesis, and (E) wound healing and tissue remodelling.

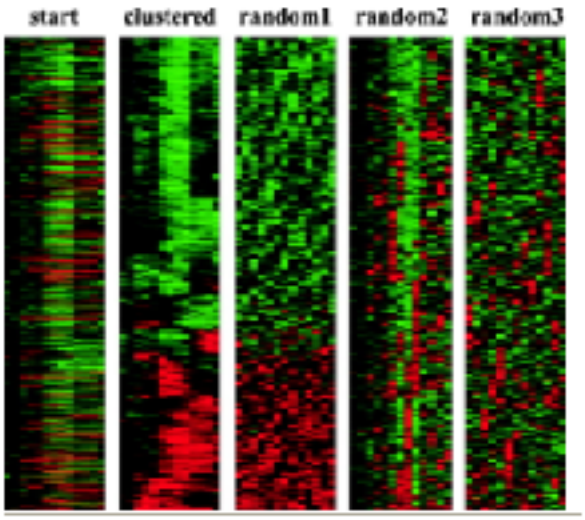

Figure 5.5: Source: [4]. To demonstrate the biological origins of patterns seen in Figure 5.4, data from Figure 5.4 was hierarchically clustered before and after random permutation within rows (random 1), within columns (random 2), and both (random 3). Indeed, the data is clustered visually, only when the "real" data was used (i.e. the "clustered" column).

5.3 Non-Hierarchal Clustering

5.3.1 K-means clustering

This method was introduced by Macqueen [8]. The idea is to partition the elements to K clusters. The heuristic moves elements between clusters if it improves the *solution cost*, E^p ,which is a function that measures the quality of the partition.

K-means clustering :

1. Initialize an arbitrary partition P into k clusters.
2. For cluster j, element $i \notin j$.
   $E^p(i,j)$ = Cost of the solution if i is moved to cluster j.
3. Pick $E^p(r,s)$ that is minimum.
4. move s to cluster r, if it improves E^p.
5. Repeat until no improvement possible.

Note that this method requires knowledge of k, the number of clusters, in advance. Since K is fixed, the algorithm aims at optimizing homogeneity, but not separation, i.e., elements in different clusters can still remain similar.
The most common use of the k-means algorithm is based on the idea of moving elements between clusters. Element v_i is moved between two clusters based on their distances to the centers of the different clusters. The solution cost function in that case is defined by:

$$E^p = \sum_p \sum_{i \in p} D(v_i, c_p)$$

Where c_p is the centroid of cluster p and $D(v_i, c_y)$ is the distance of v_i from c_y.
There are some variations of the algorithm involving changing of K. Also There are parallel versions in which we move each element to the cluster with the closest centroid simultaneously, but then convergence is not guaranteed.

5.3.2 Self organizing maps

Kohonen 1997 [6] introduced this method. Tamayo et al [14], applied it to gene expression data. Self organizing maps are constructed as follows. One chooses a grid, $m \times n$, of nodes, and a Distance function between nodes, $D(N_1, N_2)$. Each of the grid nodes are mapped into a k-dimensional space, at random. The gene vectors are mapped into the space as well. As the algorithm proceed the grid nodes are iteratively adjusted (See Figure 5.6). Each iteration involves randomly selecting a data point P and moving the grid nodes in the direction of P. The closest node n_P is moved the most, whereas other nodes are moved by smaller amounts depending on their distance from n_P in the initial geometry. In this fashion, neighboring points in the initial geometry tend to be mapped to nearby points in k-dimensional space. The process continues iteratively.

Self organizing maps :

1. Input: n-dim vector for each element (data point) p.
2. Start with a grid of $k = l \times m$ nodes, and a random n-dim associated vector $f_0(v)$ for each grid node v.
3. Iteration i :

   Pick a data point p. Find a node n_p such that $f_i(n_p)$ is the closest to p.

   Update all node vectors v as follows :

   $$f_{i+1}(v) = f_i(v) + H(D(n_p, v), i)[p - f_i(v)]$$

   Where H is a learning function which decreases with i, and decreases with $D(n_p, v)$. i.e. nodes that are not near n_p are less affected.

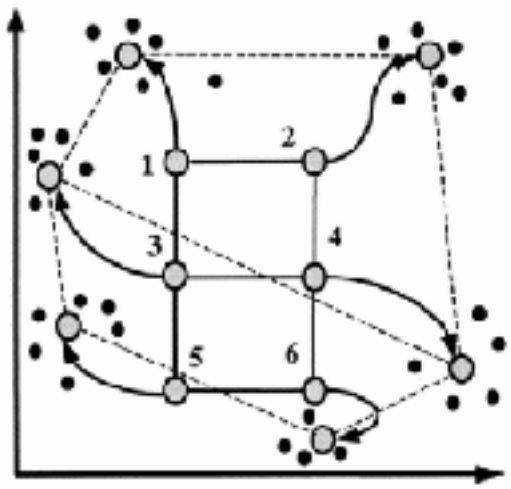

Figure 5.6: Self organizing maps : Initial geometry of nodes in a 3×2 rectangular grid is indicated by solid lines connecting the nodes. Hypothetical trajectories of nodes as they migrate to fit data during successive iterations of the self organizing maps algorithm are shown. Data points are represented by black dots, six nodes of the Self organizing map by large circles, and trajectories by arrows.

4. Repeat until no improvement possible.

The clusters are defined by the grid nodes. We assign each point (gene vector) to its nearest node n_p (cluster). Note that this method also chooses the number of clusters in advance.

GENECLUSTER

GENECLUSTER is a software that implements self organizing maps (SOM) for gene expression analysis, developed by Tamayo et al, [14]. Some results can be seen in figure 5.7. GENECLUSTER accepts an input file of expression levels from any gene-profiling method (e.g., oligonucleotide arrays or spotted cDNA arrays), together with a geometry for the nodes. The program begins with two preprocessing steps that greatly improve the ability to detect meaningful patterns. First, the data is passed through a variation filter to eliminate those genes with no significant change across the samples. This prevents nodes from being attracted to large sets of invariant genes. Second, the expression level of each gene is normalized across experiments. This focuses attention on the shape of expression patterns rather than on absolute levels of expression. A SOM is then computed. Each cluster is represented by its average expression pattern (see Figure 5.7), making it easy to discern similarities and differences among the patterns. The following learning function H(n,r,i) is used :

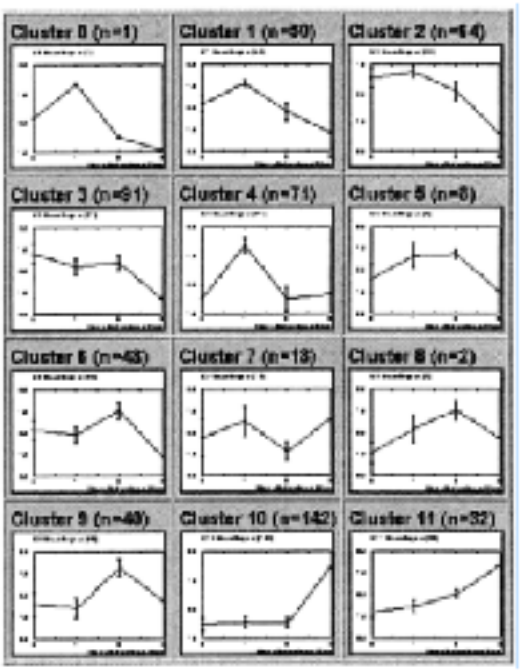

Figure 5.7: Macrophage Differentiation in HL-60 cells. The Self organizing map algorithm was applied to models of human hematopoietic differentiation. This process is largely controlled at the transcriptional level, and is related to the pathogenesis of leukemia. 567 genes were divided in to clusters using a 4x3 self organizing map. In each graph the normalized, and averaged expression levels for each cluster are shown.

$$H(n,r,i) = \begin{cases} \frac{0.02T}{T+100i} & \text{if } D(n,r) \leq \rho(i) \\ 0 & \text{otherwise} \end{cases}$$

where radius $\rho(i)$ decreases linearly with i ($\rho(0) = 3, \rho(T) = 0$). T is the maximum number of iterations, and D(n,r) denotes the distance within the grid.

5.4 Graph Clustering

The similarity matrix can be transformed into a *similarity graph*, G_θ, where the vertices are genes, and there is an edge between two vertices if their similarity is above some threshold θ, that is, $(i,j) \in E(G_\theta)$ iff $S_{i,j} > \theta$.

5.4.1 The Corrupted Clique Graph Model

The clustering problem can be modeled by a corrupted clique graph. A *clique graph* is a graph consisting of disjoint cliques. The true clustering is represented by a clique graph H (vertices are genes and cliques are clusters). Contamination errors introduced into gene expression data result in a similarity graph $C(H)$ which is not a clique graph. Under this model the problem of clustering is as follows: given $C(H)$, restore the original clique graph H and thus the true clustering.

Graph Theoretic Approach

A model for the clustering problem can be reduced to clique graph edge modification problems, stated as follows.

Problem 5.1 *Clique graph editing problem*
INPUT: $G(V, E)$ a graph.
OUTPUT: $Q(V, F)$ a clique graph which minimizes the size of the symmetrical difference between the two edge sets: $|E \Delta F|$.

Clique graph editing problem is NP-hard [12].

Problem 5.2 *Clique graph completion problem*
INPUT: $G(V, E)$ a graph.
OUTPUT: $Q(V, F)$ a clique graph with $E \subseteq F$ which minimizes $|F \setminus E|$.

The clique graph completion problem can be solved by finding all connected components of the input graph and adding all missing edges in each component. Thus the clique graph completion problem is polynomial.

Problem 5.3 *Clique graph deletion problem*
INPUT: $G(V, E)$ a graph.
OUTPUT: $Q(V, F)$ a clique graph with $F \subseteq E$ which minimizes $|E \setminus F|$.

The clique graph deletion problem is NP-hard [9]. Moreover, any constant factor approximation to the clique graph deletion problem is NP-hard as well [12].

Probabilistic Approach

Another approach is to build a probabilistic model of contamination errors and try to devise an algorithm which, given $C(H)$, reconstructs the original clique graph H with high probability.

One of the simplest probabilistic models for contamination errors is a *random corrupted clique graph*. The contamination errors are represented by randomly removing each edge in

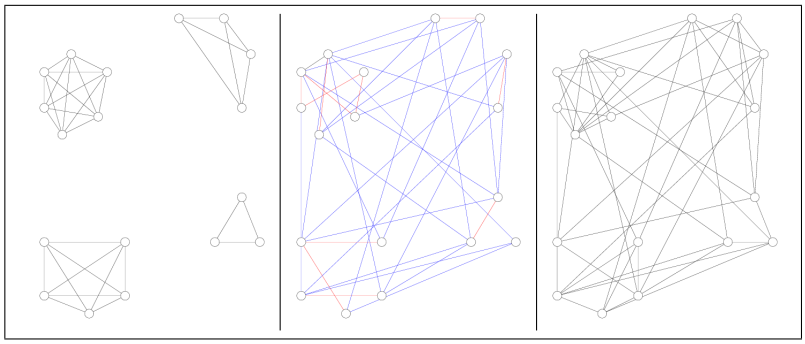

Figure 5.8: The randomly corrupted clique graph model. Left: The original Clustering H of 4 clusters, 18 elements. Middle: Random contamination (flip each edge with a probability $p < 0.5$), red edges denote edges that will be removed, blue edges denote added edges. Right: $G = C(H)$, the input (contaminated) graph.

the original clique graph H, with probability $p < 0.5$, and adding each edge not in H with the same probability, p (see Figure 5.8). We will denote by $\Omega(H, p)$ the set of all corrupted clique graphs derived from H with contamination error fraction p using this model.

5.4.2 Probabilistic Clustering Algorithm

In this section we present a clustering algorithm of Ben-Dor et. al. [2], called Parallel Classification with Cores (PPC).

We begin with a few definitions.

Definition A *cluster structure* is a vector $(s_1, ..., s_d)$, where each $s_j > 0$ and $\sum s_j = 1$. An n-vertex clique graph has structure $(s_1, ..., s_d)$ if it consists of d disjoint cliques of sizes $ns_1, ..., ns_d$.

Definition A clique graph $H(V, E)$ is called *γ-clustering* (has γ-cluster structure), if the size of each clique in H is at least $\gamma|V|$.

Definition For a fixed $0 < \gamma < 1$ we say that algorithm A *reconstructs γ-cluster structures w.h.p.* (with high probability), if for each $0 < \delta < 1$ there exists n_0 such that for each $n \geq n_0$ and for any graph $G \in \Omega(H, p)$, where H is a clique graph with n vertices and γ-cluster structure, $Prob(A(G) \neq H) < \delta$. Formally stated, $\forall \delta > 0$ $\exists n_0$ such that $\forall n \geq n_0$ and for any γ-clustering over n vertices, H, we have $Prob(A(C(H, p)) \neq H) < \delta$, where $C(H, p)$ is the random contamination of H (with p probability of flipping), and $A(C(H, p))$ is the output of algorithm A when run on input $C(H, p)$.

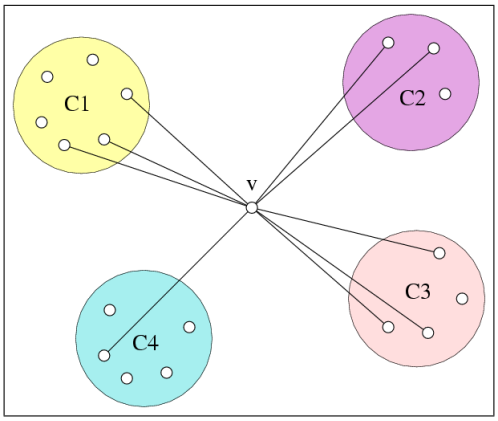

Figure 5.9: Relative Density - the highest relative density of v is with cluster C_3 (relative density of v with clusters C_1, C_2, C_3, C_4 is 1/2, 2/3, 3/4, 1/5, respectively).

Algorithm Idea

Assume that we have already clustered a subset S of vertices. Let us denote their clustering by $\{C_1, ..., C_m\}$. We will extend the clustering $\{C_1, ..., C_m\}$ to include the elements of another set S', by putting each vertex $v \in S'$ into the cluster C, to which it has the highest *relative density* (affinity), that is, the highest ratio between the number of edges connecting v to vertices in C, and the size of C (see Figure 5.9). Formally put, we choose C to be the cluster C_i which maximizes $\frac{|\{u|u\in C_i,(u,v)\in E\}|}{|C_i|}$.

After the extension $\{C_1, ..., C_m\}$ is the clustering of $S \cup S'$. Note that during the extension procedure we do not add new clusters, thus the number m of clusters is unchanged.

Later in this section, we shall show that if the clustering of S is correct then with high probability S' is also clustered correctly.

Algorithm Outline

Suppose we are given $G(V, E)$, a corrupted clique graph, that is $G \in \Omega(H, p)$ for some clique graph H with γ-cluster structure. Because H has γ-cluster structure the maximum number of cliques in H is $m = \lceil 1/\gamma \rceil$.

The PPC algorithm will perform the following steps (see Figure 5.10):

1. Uniformly draw $S_0 \subset V$, such that $|S_0| = O(\log\log(n))$;
2. Uniformly draw $S_1 \subset V \backslash S_0$, such that $|S_1| = O(\log(n))$;
3. For each clustering of S_0 into m clusters $\{C_1^0, ..., C_m^0\}$, perform:
   (a) Extend the clustering $\{C_1^0, ..., C_m^0\}$ of S_0 into clustering $\{C_1^1, ..., C_m^1\}$ of $S_0 \cup S_1$;

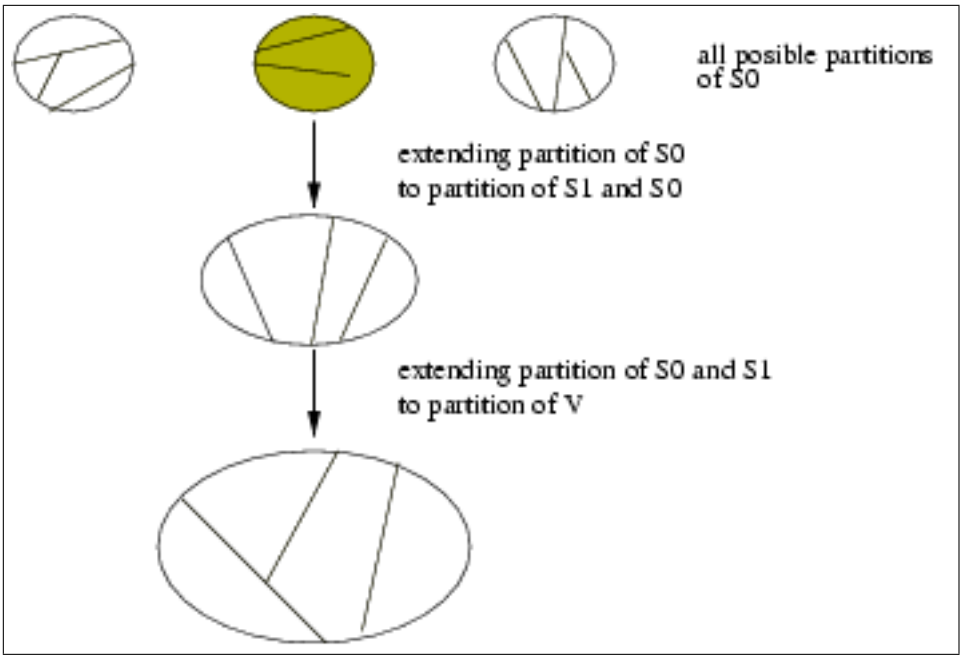

Figure 5.10: PP Calgorithm steps shown schematically.

(b) Extend the clustering $\{C_1^1, ..., C_m^1\}$ into a clustering $\{C_1, ..., C_m\}$ of V;

4. Each clustering $\{C_1, ..., C_m\}$ of V from the previous step determines a clique graph $C(V, E(C))$; from all such clusterings $\{C_1, ..., C_m\}$ of V, output the one which minimizes $|E \Delta E(C)|$;

Running Time

Running time of the algorithm is dominated by the execution of steps 3 and 4:

- Number of possible partitions of S_0 into m clusters is $m^{|S_0|} = m^{c \cdot \log\log(n)} = \log^c(n)$;
- For each partition we perform $O(\log(n)\log\log(n))$ operations in step 3a and $O(n\log(n))$ operations in step 3b, thus the overall running time of step 3 is $O(n\log^{c_1}(n))$;
- For each partition we perform $O(n^2)$ operations in step 4, thus the overall running time in step 4 is $O(n^2\log^c(n))$;
- The overall running time of the algorithm is $O(n^2\log^c(n))$;

Algorithm Correctness

Here we will give an outline of the proof. For the details of the proof please refer to [2].

Theorem 5.1 *Chernoff 1952 [3]*
Let $X \sim Binomial(n,p)$. *Let* $a < p < b$, *then:*

$$P(X \geq bn) < \exp(-nD(b\|p))$$
$$P(X \leq an) < \exp(-nD(a\|p))$$

Where $D(a\|p)$ *is the* relative entropy distance *between* $(p, 1-p)$ *and* $(a, 1-a)$, $0 < p, a < 1$, *and is defined by:*

$$D(a\|p) = p\log(\frac{p}{a}) + (1-p)\log(\frac{1-p}{1-a})$$

Theorem 5.2 *Let* H *be a clique graph with* γ*-cluster structure. For the input graph* $G(V,E) \in \Omega(H,p)$ *the output* $A(G)$ *is* H *w.h.p.*

Proof: (outline)
Consider the partition $\overline{C}^0 = \{\overline{C}^0_1, ..., \overline{C}^0_m\}$ induced by H on S_0:

- Using a simple sampling lemma it can be shown that w.h.p. each cluster of H has at least $\frac{\gamma|S_0|}{2}$ representatives in S_0, thus each non empty cluster of $\overline{C}^0$ has at least $\frac{\gamma|S_0|}{2}$ elements;
- Using the Chernoff bound it can be shown that w.h.p. we extend $\overline{C}^0$ to $\overline{C}^1$ correctly, that is $\overline{C}^1$ is induced by H on $S_0 \cup S_1$;
- Using the sampling lemma it can be shown that w.h.p. each cluster of H has at least $\frac{\gamma|S_1|}{2}$ representatives in S_1, thus each non empty cluster of $\overline{C}^1$ has at least $\frac{\gamma|S_1|}{2}$ elements (assuming that extension from $\overline{C}^0$ to $\overline{C}^1$ was done correctly);
- Using the Chernoff bound it can be shown that w.h.p. we extend $\overline{C}^1$ to $\overline{C}$ correctly, that is $\overline{C}$ is induced by H on V;
- Thus in step 4 of the algorithm we have w.h.p. a correct clustering. To complete the proof we must show that w.h.p. for each other partition C that we have in step 4, $|E\Delta E(\overline{C})| < |E\Delta E(C)|$;

■

Below we will show in some detail a result, which proves the last statement of the above outline.

Theorem 5.3 *Let H be a clique graph with a γ-cluster structure. Then w.h.p. for any $G \in \Omega(H,p)$, H is the closest clique graph to G. That is, for any other clique graph C with a γ-cluster structure, the following holds w.h.p.: $|E(H)\Delta E(G)| < |E(C)\Delta E(G)|$.*

Proof: (outline)
First we will show that given some random clique graph C with a γ-cluster structure we have w.h.p. $|E(H)\Delta E(G)| < |E(C)\Delta E(G)|$:

- A key observation is that C is closer to G iff more than half of the edges in $E(H)\Delta E(C)$ were flipped when generating graph G (please refer to Figure 5.11 for a schematic proof);
- $P(|E(C)\Delta E(G)| < |E(H)\Delta E(G)|) = P(\text{number of edges flipped} \geq \frac{|E(H)\Delta E(C)|}{2})$;
- Using the Chernoff bound it can be shown that $P(|E(C)\Delta E(G)| < |E(H)\Delta E(G)|) < \exp(-|E(H)\Delta E(C)|D(0.5\|p))$;

Now we have to show that for any clique graph C with a γ-cluster structure we have w.h.p. $|E(H)\Delta E(G)| < |E(C)\Delta E(G)|$.

- Consider all clique graphs C with a γ-cluster structure, such that $|E(C)\Delta E(H)| \geq \frac{n\log(n)}{D(0.5\|p)}$. The number of such graphs is m^n where $m = \lceil 1/\gamma \rceil$. Thus, using previous results, it can be easily shown that for any such C w.h.p. $|E(H)\Delta E(G)| < |E(C)\Delta E(G)|$;
- Consider all clique graphs C with a γ-cluster structure, such that $|E(C)\Delta E(H)| < \frac{n\log(n)}{D(0.5\|p)}$, a similar but more complicated argument shows that w.h.p. for any such C $|E(H)\Delta E(G)| < |E(C)\Delta E(G)|$;

■

5.4.3 Practical Heuristic - Algorithm CAST

Although the theoretical ideas presented in the previous section show asymptotic running time complexity of $O(n^2 \log^c n)$, their implementation is still impractical (the constants, for instance, are very large, as in the computation of all possible partitions of S_o into m clusters in step 3). Therefore, based on ideas of the theoretical algorithm, CAST (Cluster Affinity Search Technique), a simple and practical heuristic, was developed. All the tests described in subsequent sections were performed using this practical implementation of the theoretical algorithm.

Let C be a cluster. Let $S_{i,j}$ be a similarity matrix and let $v \in V$ be a gene. We define the *affinity of v to cluster C* by $\frac{\sum_{u \in C} S_{u,v}}{|C|}$. Given an *affinity threshold* τ we will say that v is

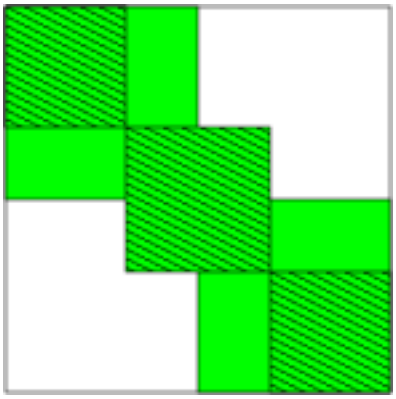

Figure 5.11: Schematic proof of the key observation in Theorem 5.3. This figure depicts a graph G (blank), an original clique graph H (green) and some other clique graph C (green with pattern). Consider $E(H)\Delta E(G)$ and $E(C)\Delta E(G)$. Each edge (u, v) which belongs to neither C nor to H, but belongs to G increments both $E(H)\Delta E(G)$ and $E(C)\Delta E(G)$. Each edge (u, v) which belongs to both C and H, but does not belong to G also increments both $E(H)\Delta E(G)$ and $E(C)\Delta E(G)$. Thus, the difference between $E(H)\Delta E(G)$ and $E(C)\Delta E(G)$ is due to patches that correspond to $E(H)\Delta E(C)$. Let us denote by Δ^* the symmetric difference constrained to edges in $E(C)\Delta E(H)$. Clearly $|E(C)\Delta^* E(G)| + |E(H)\Delta^* E(G)| = |E(H)\Delta E(C)|$. Thus $|E(C)\Delta^* E(G)| < |E(H)\Delta^* E(G)|$ iff $|E(H)\Delta^* E(G)| > \frac{|E(H)\Delta E(C)|}{2}$. Hence, $|E(C)\Delta E(G)| < |E(H)\Delta E(G)|$ iff $|E(H)\Delta^* E(G)| > \frac{|E(H)\Delta E(C)|}{2}$.

a close gene to cluster C if its affinity to C is above τ and we will say that v *is a weak gene in* C if its affinity to C is below τ.

Following are the steps of the practical implementation:

- Construct one cluster at a time and denote it by CC;
- At each step either:
  - Add a close gene to CC;
  - Remove a weak gene from CC;
  - Close CC when no addition or removal is possible;
- Repeat until all genes are clustered;

The main differences between the practical implementation and the theoretical algorithm are:

- In the theoretical algorithm several partitions are formed and then the "best" partition is chosen; in the practical implementation one partition is formed by building one cluster at a time.

- The theoretical algorithm considers the similarity graph, while the practical implementation processes the similarity matrix (the similarity value between any two genes can assume any real value).

- In the theoretical algorithm, the clusters in a partition are extended by adding new elements to them; the practical implementation also allows to remove a weak element from a cluster.

Although little can be proved about the running time and performance of the practical implementation, the test results described in the next sections show that it performs remarkably well, both on simulated data and on real biological data.

BioClust

BioClust is an implementation package of the CAST heuristic. The following section presents results of applying BioClust on both synthetic data and real gene expression data.

Clustering Synthetic Data

The simulation procedure is as follows (please refer to Figure 5.12 for visualization of the simulation procedure):

- Let H be the original clique graph.

- Generate G from H by independently removing each edge in H with probability p and adding each edge not in H with probability p.

- Randomly permute the order of vertices in G and run BioClust with affinity threshold $\tau = 0.5$.

- Compare BioClust's output to the original graph H.

There are several comparison criteria, which can be used to compare the algorithm's output to the original clique graph. Given two adjacency matrices A and B of two graphs of the same size, let N_{ij} be the number of entries on which A and B have values i and j, respectively. The *matching coefficient* is defined by $\frac{N_{00}+N_{11}}{N_{00}+N_{01}+N_{10}+N_{11}}$ that is total number of matching entries divided by total number of entries. The *Jaccard coefficient* is defined by $\frac{N_{11}}{N_{01}+N_{10}+N_{11}}$, which is similar to the matching coefficient, only with N_{00}, the number of entries which are zero in both matrices, removed. In sparse graphs N_{00} will be a dominant factor, thus Jaccard coefficient is more sensitive when dealing with sparse graphs. With both coefficients, the higher the value, the closer the result is to the real clustering. Both coefficients have maximum value of 1, which implies perfect clustering.

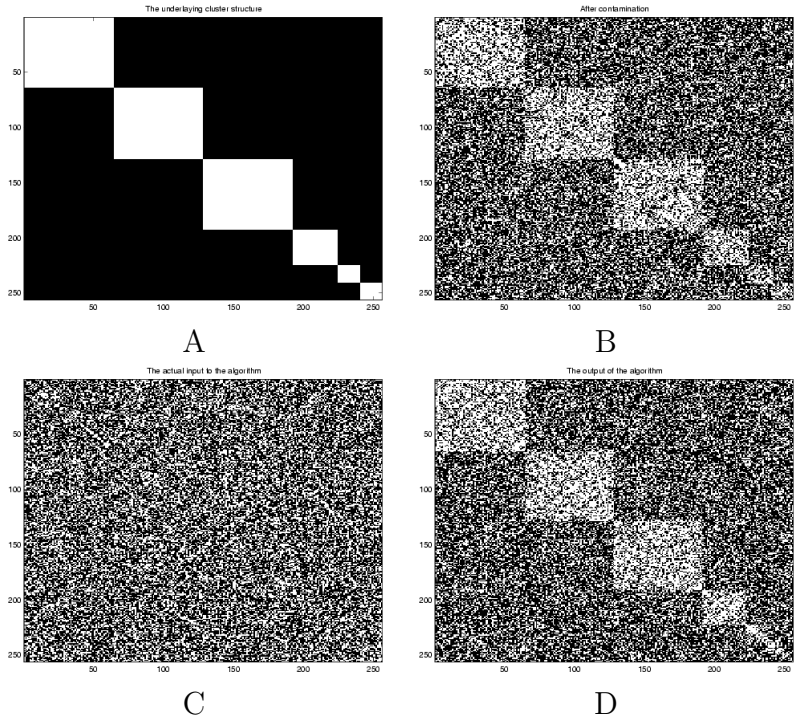

Figure 5.12: Source: [2]. A visualization of the simulation procedure. **A**: The adjacency matrix of the original clique graph H before introduction of errors. Position (i, j) is white if $(i, j) \in E(H)$, that is, if i and j belong to the same cluster. **B**: The same matrix after introduction of errors. Note that the cluster structure is still visible for all but the smallest clusters. **C**: The same as **B** but vertex order is randomly permuted. This is the actual input to the algorithm. **D**: Matrix **C** reordered according to solution produced by the algorithm. With the exception of perhaps the smallest clusters, the essential cluster structure is reconstructed.

cluster structure	n	p	matching coeff.	Jaccard coeff.
$\{0.4, 0.2, 0.1 \times 4\}$	500	0.2	1.0	1.0
$\{0.4, 0.2, 0.1 \times 4\}$	500	0.3	0.999	0.995
$\{0.4, 0.2, 0.1 \times 4\}$	500	0.4	0.939	0.775
$\{0.1 \times 10\}$	1000	0.3	1.0	1.0
$\{0.1 \times 10\}$	1000	0.35	0.994	0.943

Table 5.1: Performance of BioClust for different values of p and n. Mean values of matching coefficient and Jaccard coefficient are given.

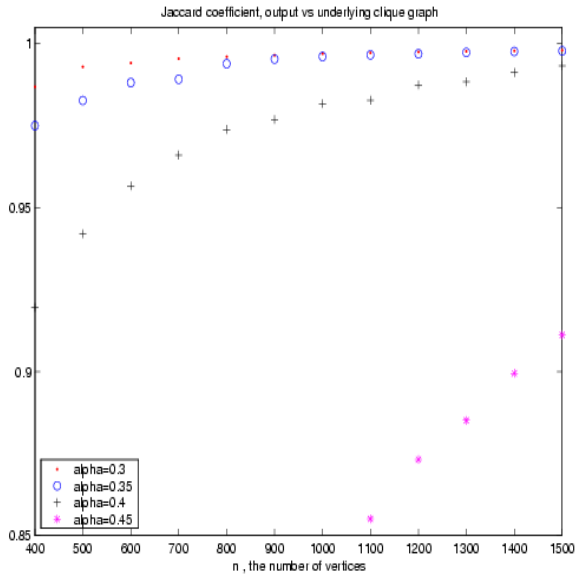

Figure 5.13: Source: [2]. Simulation results for H with cluster structure of $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}\}$. The x-axis is n, the number of vertices in H (clustered entities), and y-axis is the mean value of the Jaccard coefficient. Each curve corresponds to a specific probability $p = \alpha$ of contamination error.

Table 5.1 presents results of simulation for different values of contamination error p and/or number of cluster entities n. The values of the matching coefficient and the Jaccard coefficient are presented. It can be seen that the Jaccard coefficient is more sensitive. One can also observe the effect of p and n on the performance of the algorithm.

Figure 5.13 presents results of simulations for different values of n and p. It can be seen that the properties of the theoretical algorithm are preserved in its practical implementation. We get better performance when the number of clustered entities (vertices in H) increases.

Clustering Temporal Gene Expression Data

The gene expression data used in this experiment is from [15]. In this paper the authors study the relationship among expression patterns of genes involved in the rat Central Nervous System (CNS).

Gene expression patterns were measured for 112 genes along 9 different development time points. The gene expression data for each gene was augmented with derivative values to enhance the similarity for closely parallel but offset expression patterns, resulting in a 112×17 expression matrix. The similarity matrix was obtained using Euclidean distance. The execution of BioClust resulted in eight clusters. Since partitioning to clusters is known from [15] this experiment was done mainly for validation of the algorithm.

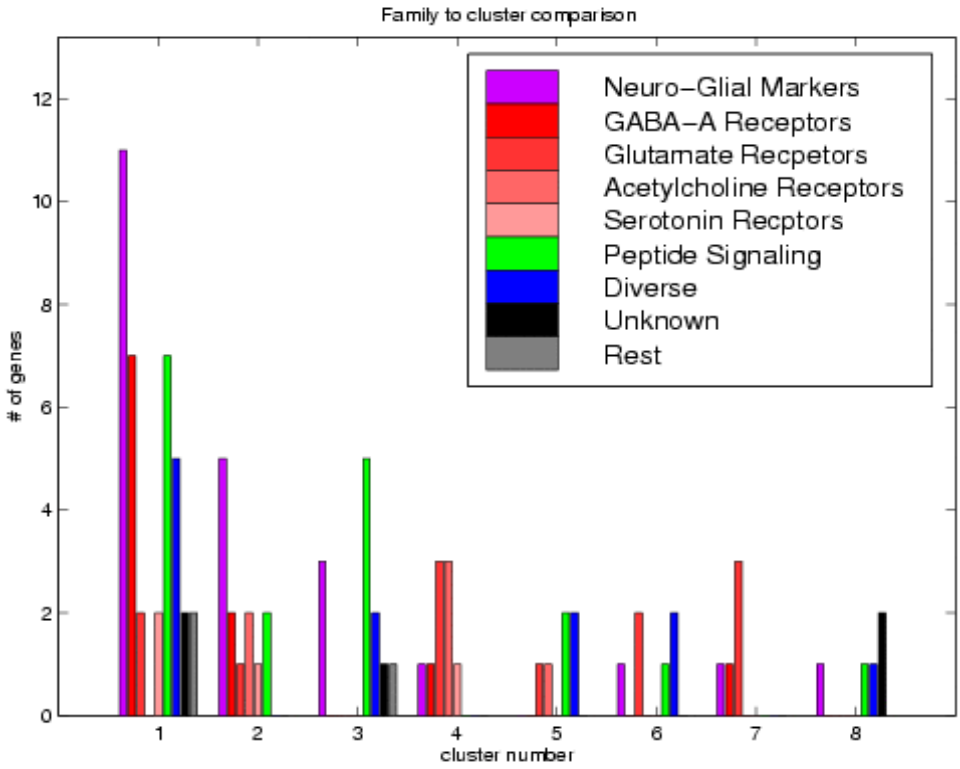

Figure 5.14: Source: [2]. Applying the algorithm to temporal gene expression data [15]. The solution generated by the algorithm is compared to the prior classification. For each cluster (x-axis), bars composition in terms of biologically defined families. The height of each bar (y-axis) represents the number of genes of a specific cluster family. Most clusters contain predominantly genes from one or two families.

Figures 5.14 and 5.15 present the clustering results. Note that all clusters, perhaps with the exception of cluster #1, manifest clear and distinct expression patterns. Moreover, the agreement with the prior biological classification is quite good.

Clustering C. Elegans Gene Expression Data

The gene expression data used in this analysis is from [5]. Kim et al. studied gene regulation mechanisms in the nematode C. Elegans. Gene expression patterns were measured for 1246 genes in 146 experiments, resulting in a 1246×146 expression matrix. The similarity matrix was obtained using Pearson correlation.

The algorithm found 40 clusters. Only very few genes out of the 1246 were classified into families by prior biological studies. The algorithm clustered these families quite well into few homogeneous clusters (see Figure 5.16).

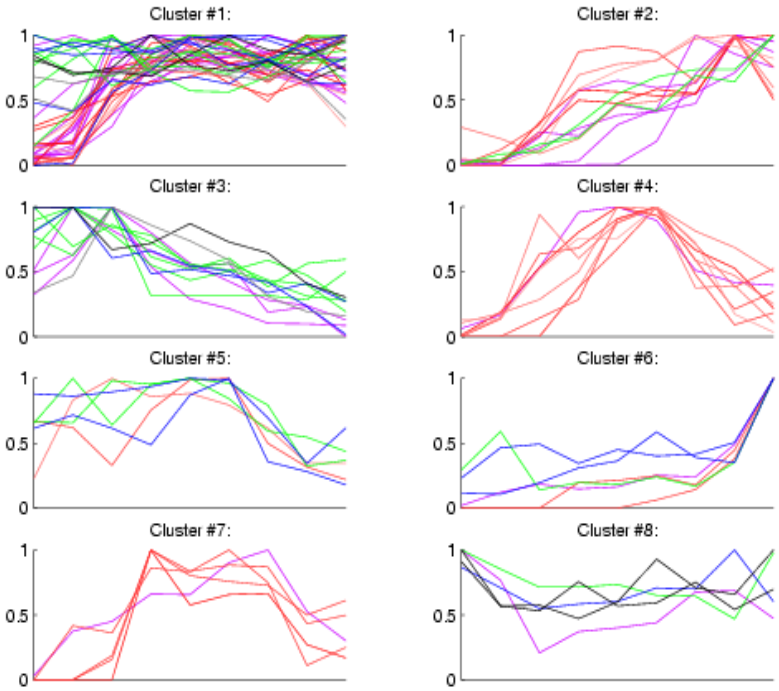

Figure 5.15: Source: [2]. Applying CAST to temporal gene expression data [15]. Each graph presents expression patterns of genes in a specific cluster. The x-axis represents time, while the y-axis represents normalized expression level.

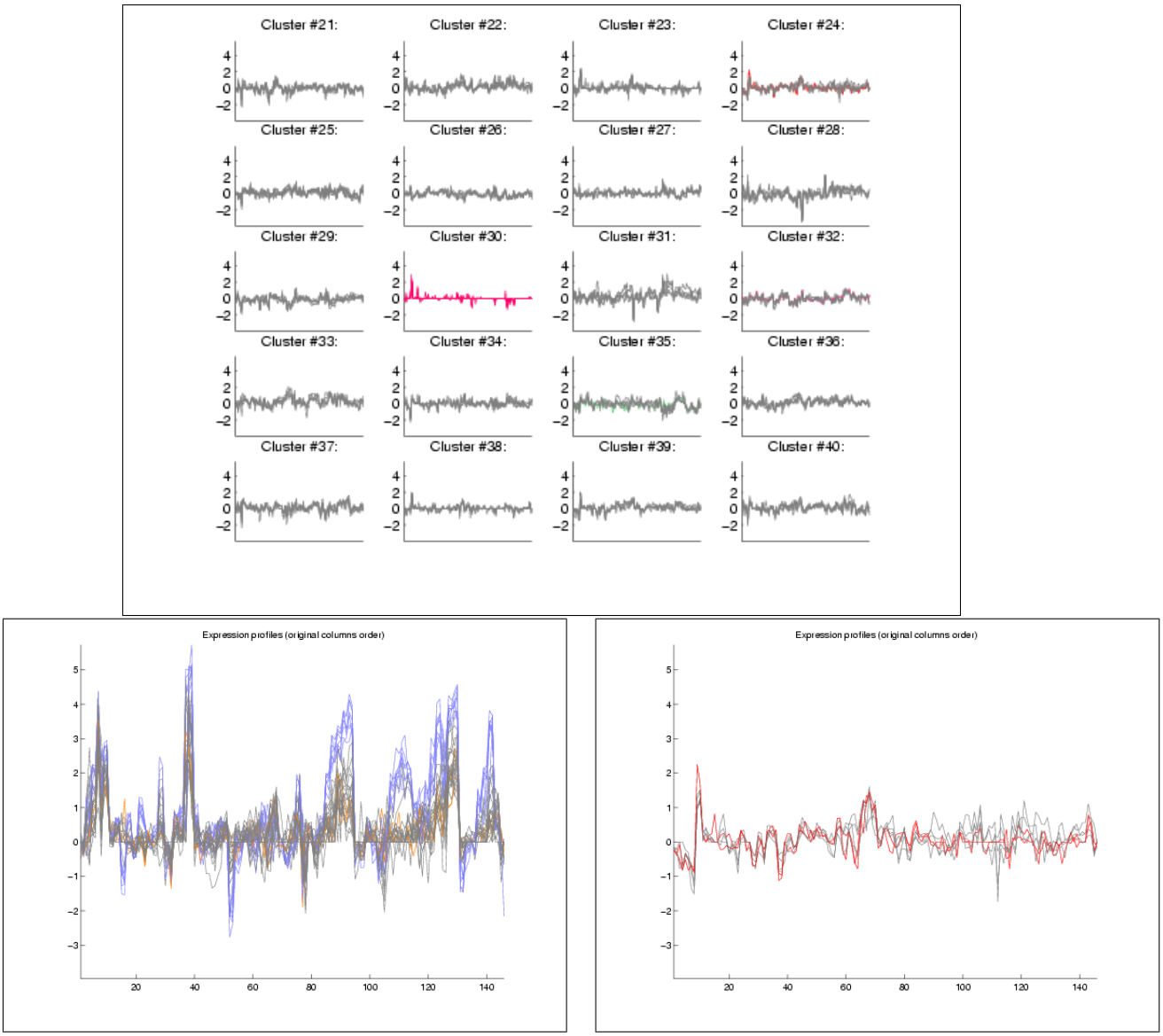

Figure 5.16: Source: [2]. Some results of the CAST algorithm applied to the nematode gene expression data of Kim et. al. Top: expression patterns for clusters #21 to #40. x axis: conditions (matrix columns) in arbitrary order. y axis: intensity level. Most of the genes' functions are unknown, so only few genes are color coded. Blue: sperm genes; red: yeast genes (control) ; gray: unknown. Note the homogeneity of cluster #30. Bottom Left: expression patterns of the genes in cluster #1, consisting of 31 genes. Bottom Right: Expression patterns of the six genes in cluster #24. This cluster contains two growth related genes, lin15 and E2F. This suggests the hypothesis that the other four members of this cluster have related functions.

One example of the potential use of clustering for analyzing gene expression patterns is shown in Figure 5.16. A six-gene cluster (cluster #24) contained two growth-related genes and four anonymous genes. This suggests the possibility that the other four genes are also growth-related, paving the way for future biological research.

Tissue Clustering

The gene expression data used in this experiment is from [1]. The authors describe an analysis of gene expression data obtained from 62 samples of colon tissue, 40 tumor and 22 normal tissues. Gene expression patterns were measured for 2000 genes in the 62 samples, using an Affymetrix chip. The similarity between each two samples was measured using Pearson correlation. Note that here, the similarity is measured between tissues, not genes.

BioClust formed 6 clusters of the data. Figure 5.17 shows the distribution of tumor and normal tissues in the six clusters produced.

The main goal of clustering here is to achieve a separation of tumor and normal tissues. This experiment demonstrates the usefulness of clustering techniques in learning more about the relationship of expression profiles to tissue types.

Improved Theoretical Results

In [11], Tsur & Shamir have introduced a generalized random clique graph model with improved theoretical results, including reduction of the $\Omega(n)$ restriction on cluster sizes, and stronger results when cluster sizes are almost equal.

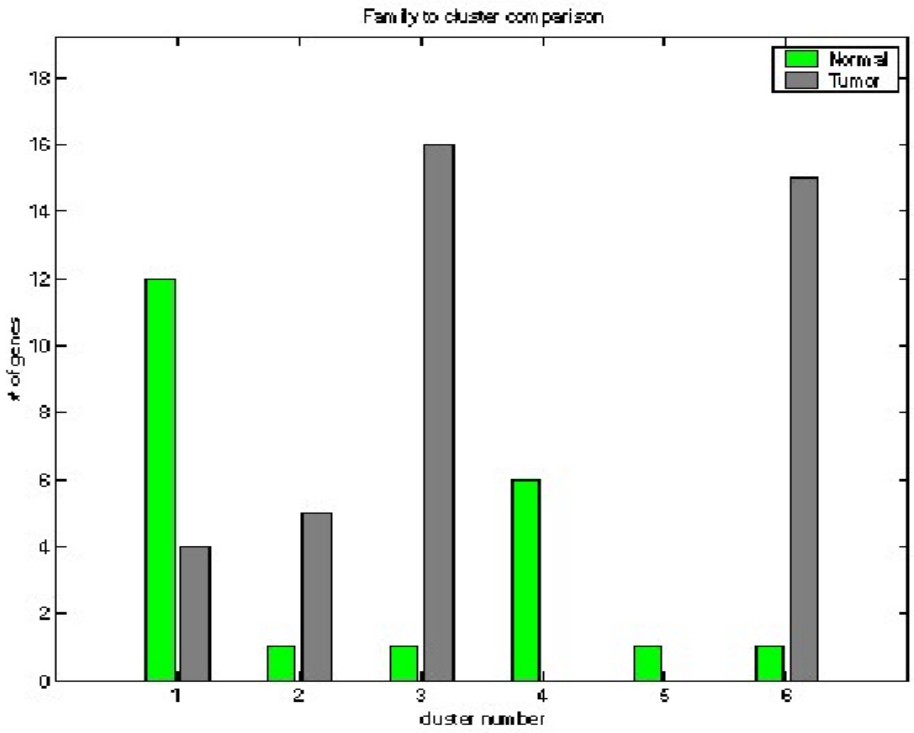

Figure 5.17: Source: [2]. Distribution of tumor and normal tissues in the six clusters produced by the CAST algorithm.

Bibliography

[1] U. Alon, N. Barkai, D. A. Notterman, G. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS*, 96:6745–6750, June 1999.

[2] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[3] H. Chernoff. A measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[4] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[5] S. Kim. Department of Developmental Biology, Stanform University, `http://cmgm.stanford.edu/~kimlab/`.

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

[7] G.N. Lance and W.T. Williams. A general theory of classification sorting strategies. 1. hierarchical systems. *The computer journal*, 9:373–380, 1967.

[8] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1965.

[9] A. Natanzon. Complexity and approximation of some graph modification problems. Master's thesis, Department of Computer Science, Tel Aviv University, 1999.

[10] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[11] R. Shamir and D. Tsur. Improved algorithms for the random cluster graph model. In *Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT '02)*, LNCS 2368, pages 230–239. Springer-Verlag, 2002.

[12] R. Sharan, R. Shamir, and D. Tsur. Cluster graph modification problems. In *Proc. 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '02)*, To appear.

[13] R. Sokal and C. Michener. A statistical method for evaluating systematic relationships. *Univeristy of Kansas Science Bulletin*, 38:1409–1438, 1958.

[14] 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*, 96:2907–2912, 1999.

[15] 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. *PNAS*, 95(1):334–339, 1998.