Analysis of Gene Expression Data **Spring Semester, 2002**

Lecture 4: April 25, 2002

Lecturer: Ron Shamir *Scribe: Dror Fidler and Shahar Harrusi*[1]

4.1 Introduction

4.1.1 Functional Genomics

Having (almost) 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. Functional annotation of genes is still in its early stages. For example, in the plant Arabidopsis (whose sequencing has recently been completed), there is no functional annotation for over 40% of the genes. 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.

4.1.2 Gene Expression

The holistic approach to functional genomics relies ,among others, on high-throughput methods for measuring gene expression. This is basically the amount of protein (coded for by the gene) present in a cell at a specific moment. The existing methods for measuring gene expression are based on two biological assumptions:

[1]Based in part on a scribe by Ronny Morad and Tal Moran, Algorithms in Molecular Biology, Fall 2001.

1. *Transcription level of genes indicates their regulation:* Since protein is generated from a gene in a number of stages (transcription, splicing, synthesis of protein from mRNA), regulation of gene expression can occur at many points. This assumption means that most regulation is done only during the transcription phase.

2. *Only genes which contribute to organism fitness are expressed :* This means that genes which are irrelevant to the organism's function are not expressed.

Relying on these assumptions, we can conclude that detecting changes in gene expression level provides clues on the function of its product.

4.1.3 DNA Chips/Microarrays

As was explained in lecture 1 and lecture 2, there are two basic types of DNA chips, called *format I*, and *format II*, the difference being whether the target DNA is on the chip (format I) or in the "air" (format II). There are several variants on these 2 basic types :

1. *Oligonucleotide arrays* :
   format II. Short oligo sequences are on the chip.

2. *cDNA microarrays* :
   format II. Gene specific cDNA's are on the chip.

3. *Oligo-fingerprinting* :
   format I. The first type of chip that was used.

4.1.4 cDNA Clustering

As we have seen, genes affect the cell by being *expressed*, i.e., transcribed into mRNA and translated into proteins that react with other molecules. It is therefore highly interesting to analyze the *expression profile* of genes, i.e., in which tissues and at what stages of development they are expressed. From this information we can sometimes guess what the functions of these genes are. This is especially true if we discover that the expression profile of an unknown gene is similar to that of a known gene. Usually, in such cases the functions of these genes are related. Another important piece of information is the level of expression of each gene. Different genes have different levels of expression,indicating their activity level in the cell. It is therefore highly informative to find which genes are expressed in each tissue, and in what level.

This is easier said than done. An average tissue contains more than 10,000 expressed genes, and their expression levels can vary by a factor of 10,000. Therefore, in order to make sure we detect all the genes in a tissue, we should extract more than 10^5 transcripts per tissue.

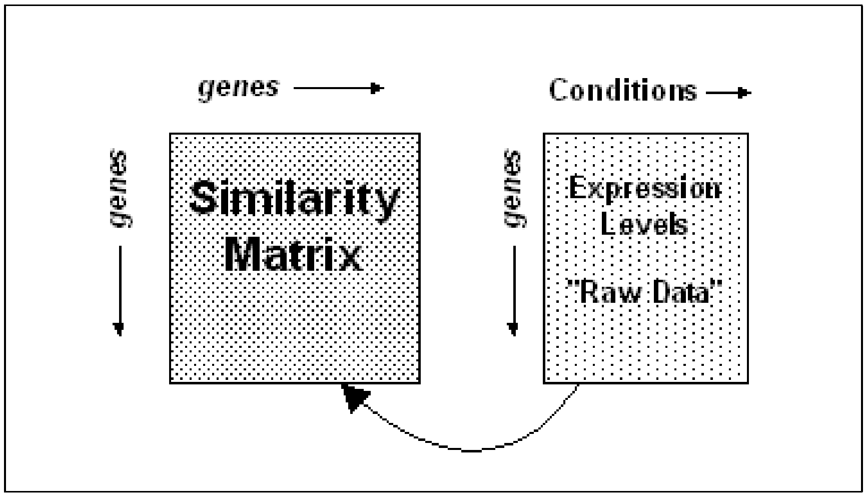

Figure 4.1: The raw data matrix maps conditions with gene expression. The *Similarity matrix* is derived from the raw data matrix, according to a similarity, or distance function.

Keeping in mind that there are about 100 different types of tissue in the body, and that we are interested in comparing different growth stages (or disease stages), we can conclude that we should analyze more than 10^{10} transcripts. Therefore sequencing all the cDNAs is expensive and wasteful. Obviously, we need cheap, efficient and large scale methods.

4.2 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. an Experiment is a set of conditions. Thus, R_{ij} = is the expression level for gene i, at condition j. The expression data can represent ratios, absolute values, or distributions.

In order to analyze the data, it is preprocessed to compute a *similarity* matrix, or a *distance* matrix. Note that the similarity matrix is larger than the raw data matrix since there are usually much more genes than conditions. Figure 4.1 shows the data and similarity matrices.

4.3 Some DNA chip applications

- Deducing functions of unknown genes (similar expression pattern implies similar function).

- Identifying disease profiles.
- Deciphering regulatory mechanisms (co-expression implies co-regulation).
- Classification of biological conditions.
- Genotyping.
- Drug development.

A first step in most of these analysis is clustering genes or conditions based on their expression levels.

4.4 Clustering

Clustering methods have been used in a vast number of fields. The goal in a clustering problem is to group elements (genes) to clusters satisfying:

1. *Homogeneity* : Elements inside a cluster are highly similar to each other.
2. *Separation* : Elements from different clusters have low similarity to each other.

We can distinct 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.

There are several encouraging facts resulting from clustering research :

- Distinct measurements of same genes cluster together.
- Genes of similar function cluster together.
- Many cluster-function specific insights are gained.

4.4.1 Hierarchic Clustering

This alternative 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 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.

4.4.2 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 4.3).

4.4.3 Neighbor Joining Algorithm

A simple algorithm, based on neighbor merging, is due to Saitou and Nei [13]. 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 $D_{rs} = \min_{ij}(D_{ij})$.

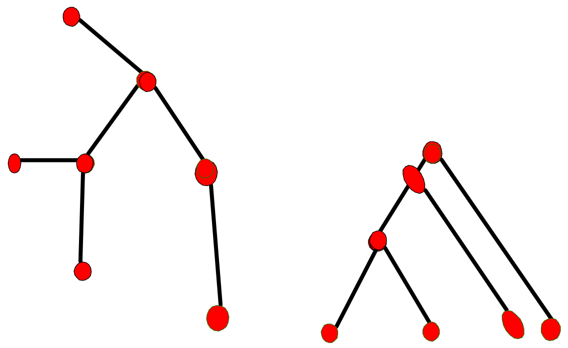

Figure 4.2: Hierarchal data can be represented as a rooted or un rooted tree

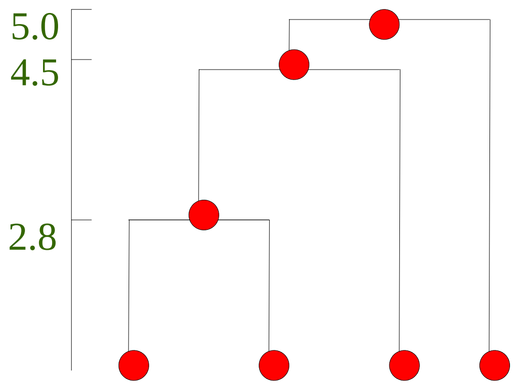

Figure 4.3: In a dendrogram, distances are represented on the y-axis. Denote the leaves a, b, c, d (from left to right). Then $D(a, b) = 2.8$, $D(a, c) = D(b, c) = 4.5$, $D(b, d) = D(c, d) = 5.0$

3. Merge clusters r,s.

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.

4.4.4 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 [8], and Sokal and Michener [14].

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 :

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

4.4.5 A General Framework

Lance and Williams, [8] 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 :

$$\gamma = 0$$

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

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

4.4.6 Hierarchical clustering of gene expression data

A series of experiments were performed on real gene expression data, by Eisen et al. [3] 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. The original data of test to reference ratios was first log transformed, and then normalized, to have mean 0 and variance 1. Denote by N_{ij} 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, and the tree is presented by ordering the leaves according to increasing subtree weight (see Figure 4.4). The weight can be average expression level, time of maximal induction, or any other. Figure 4.5 demonstrates the different output given by hierarchical when clustering gene expression data, and random data.

4.5 Non-Hierarchal Clustering

4.5.1 K-means clustering

This method was introduced by Macqueen [10]. 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.

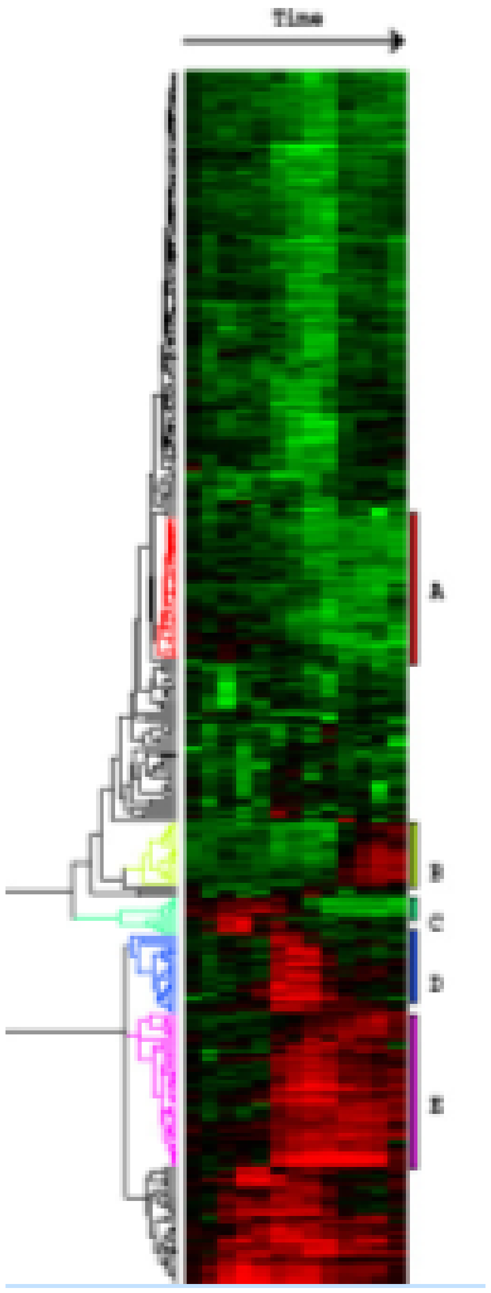

Figure 4.4: Source: [3]. The dendrogram resulting from the starved human fibroblast cells experiment. five major clusters can be seen, and many non clustered genes. The genes in the five groups serve similar functions : (A) cholesterol biosynthesis, (B) the cell cycle, (C) the immediate-early response, (D) signaling and angiogenesis, and (E) wound healing and tissue remodelling.

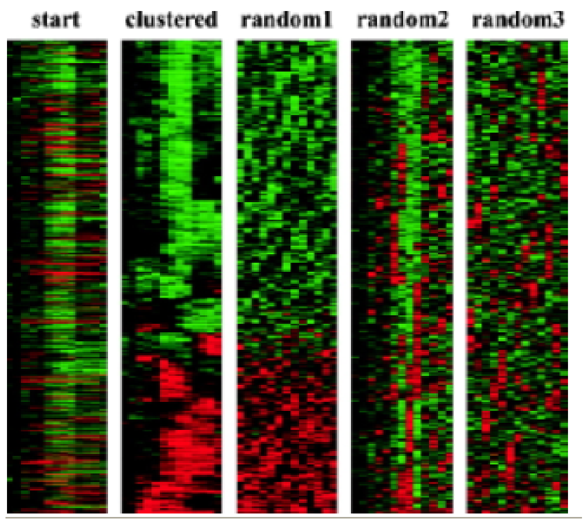

Figure 4.5: Source: [3]. To demonstrate the biological origins of patterns seen in Figure 4.4, data from Figure 4.4 was hierarchically clustered before and after random permutation within rows (random 1), within columns (random 2), and both (random 3). Indeed, the data is clustered visually, only when the "real" data was used (i.e. the "clustered" column).

4.5.2 K-means variations

The k-means variations algorithm is based on the idea of moving elements between clusters, based on their distances to the centers of the different clusters. Since K is fixed, it aims at optimizing homogeneity, but not separation. i.e., elements in different clusters can still remain similar. There are some variations for changing K. There is also a parallel version in which we move each element to the cluster with the closest centroid simultaneously, but then convergence is not guaranteed.

K-means variations :

1. Input: vector v_i for each element i.
2. Compute a centroid c_p for each cluster p, e.g., gravity center = average vector.
3. Compute the solution cost:
$$E^p = \sum_p \sum_{i \in p} D(v_i, c_p)$$
Where $D(v_i, c_y)$ is the distance of v_i from c_y.
4. Compute $E^p(i,j)$ = change in solution cost if i is moved to cluster j.

5. Perform the best improvement to the solution cost.

6. Repeat until no improvement possible.

4.5.3 Self organizing maps

Kohonen 1997 [7] introduced this method. Tamayo et al [16], applied it to gene expression data. Self organizing maps are constructed as follows. One chooses a grid of nodes, and a Distance function between nodes, $D(N_1, N_2)$. The nodes are mapped into k-dimensional space, initially at random, and then iteratively adjusted (See Figure 4.6). Each iteration involves randomly selecting a data point P and moving the 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.

4. Repeat until no improvement possible.

4.5.4 GENECLUSTER

GENECLUSTER is a software that implements self organizing maps (SOM) for gene expression analysis, developed by Tamayo et al, [16]. Some results can be seen in figure 4.7. 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

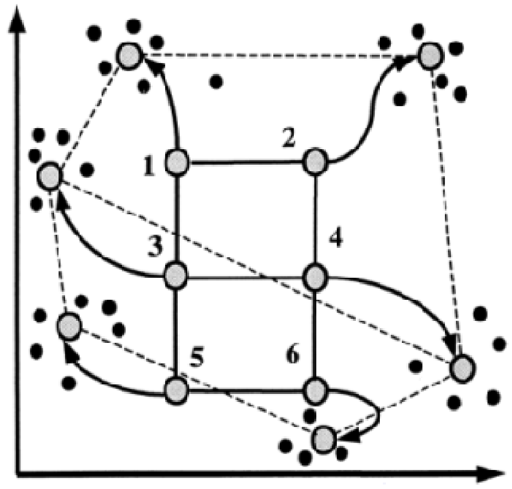

Figure 4.6: Self organizing maps : Initial geometry of nodes in a 3×2 rectangular grid is indicated by solid lines connecting the nodes. Hypothetical trajectories of nodes as they migrate to fit data during successive iterations of the self organizing maps algorithm are shown. Data points are represented by black dots, six nodes of the Self organizing map by large circles, and trajectories by arrows.

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. An SOM is then computed. Each cluster is represented by its average expression pattern (see Figure 4.7), making it easy to discern similarities and differences among the patterns. The following learning function H(n,r,i) is used :

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

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.

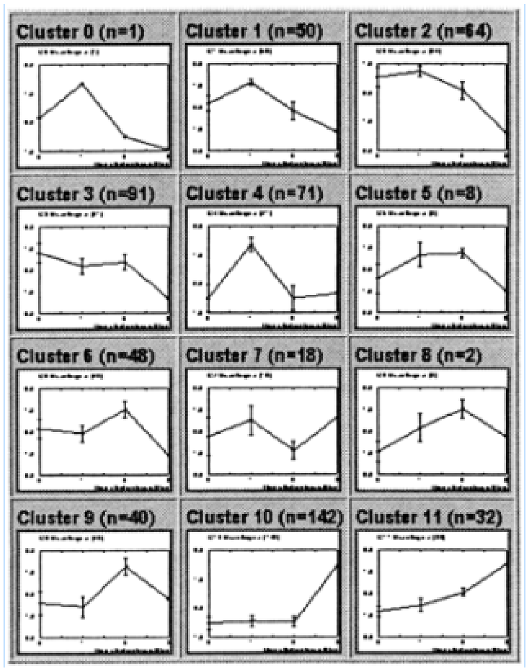

Figure 4.7: Macrophage Differentiation in HL-60 cells. The Self organizing map algorithm was applied to models of human hematopoietic differentiation. This process is largely controlled at the transcriptional level, and is related to the pathogenesis of leukemia. 567 genes were divided in to clusters using a 4x3 self organizing map. In each graph the normalized, and averaged expression levels for each cluster are shown.

4.6 Clustering cDNA oligo-fingerprints

4.6.1 Oligo-fingerprinting

This technique was developed by several researchers: Drmanac-Crkvenjakov [2], Bains-Smith [1], Southern [15] and Macevics [9].

In this technique the targets (fragments of cDNA) are placed at high density on an array/membrane. We pick a set of short oligos, and hybridize each oligo to the membrane. Thus we obtain a binary fingerprint vector f for each target: $f_i = 1$ iff oligo i hybridized to the target. Many cDNA's extracted from a tissue belong to the same gene. (About 100,000 fragments of cDNA are extracted, each one between 500 - 2500 long). If 2 fragments of cDNA belong to the same gene then they have a common *3‘ end.* (Each fragment of DNA is chemically oriented, one side is called 5 prime (5‘), and the other 3 prime (3‘)). Hence, cDNAs of the same gene have similar fingerprints. (Since they have a common 3‘ end). This type of data motivates the following approach to detecting the identity and level of cDNA‘s in a given tissue :

1. Cluster cDNAs according to their fingerprints.
2. Sequence representatives from each cluster, to obtain a sequence that identifies the gene.

4.6.2 The HCS Algorithm

The algorithm we will describe here is due to Hartuv et al. [5]. Let us define a graph $G = (V, E)$, where the vertices represent the spotted cDNAs, and an edge $e = (v_1, v_2)$ exists if v_1 and v_2 have similar fingerprints, that is, their similarity is above a certain threshold. Recall the following definitions:

1. The *connectivity* $k(G)$ of a graph G is the minimum number of edges whose removal results in a disconnected graph. If $k(G) = l$ then G is said to be *l-(edge)-connected.*
2. A *cut* in G is a set of edges who removal disconnects the graph. A *minimum cut* is a cut with minimum number of edges. If C is a minimum cut set of a non-trivial graph G, then $|C| = k(G)$. Hence, a k-connected graph is a nontrivial graph in which the size of a minimum cut is k.

Had the similarity graph perfectly represented the cluster structure, each cluster would have formed a clique, as all members of a cluster are highly similar, and no two clusters would have been connected by an edge. In practice, searching for cliques in the graph would fail on two accounts: First, finding maximum cliques is computationally intractable [4]. Second, and more important, real data matrices (and cDNA hybridization matrices in particular)

```
Procedure HCS($G(V, E)$)
    ($H$,$\bar{H}$, $cut$) $\leftarrow$ min-cut($G$)
    **if** ($|cut| \geq \frac{|V|}{2}$)
        **return** $V$
    **else**
        HCS($H$)
        HCS($\bar{H}$)
    **end if**
**end**
```

Figure 4.8: The basic HCS algorithm.

contain many errors. In terms of the similarity graph, false negatives correspond to missing edges between vertices in the same cluster, while false positive errors correspond to extra edges between vertices of different clusters.

In cDNA fingerprinting, errors in the hybridization data generate inexact fingerprinting, leading in turn to errors in the similarity graph. That error rate is very high: The false negative rate in the similarity graph is above 50% and the false positive rate is smaller but still significant (especially since the true graph has much more non-edges than edges).

A key definition for our approach is the following: A graph G with $n > 1$ vertices is called *highly connected* if $k(G) > \frac{n}{2}$. A *highly connected subgraph* (HCS) is an induced subgraph $H \subseteq G$ such that H is highly connected. The HCS algorithm identifies highly connected components of a given input graph. The algorithm is given in figure 4.8. It is assumed that the procedure *min-cut*(G) returns H, $\bar{H}$ and C, where C is a minimum cut set which separates G into the subgraphs H and $\bar{H}$.

It works as follows: In each iteration, it finds the minimum cut in the graph, and separates the graph into two subgraphs. If the current graph is highly connected, the algorithm stops (as it has found a cluster). Otherwise, it recursively continues processing each of the two subgraphs.

Running Time

The algorithm runs in $O(N \times f(n, m))$, where N is the number of clusters and $f(n, m)$ is the time to find the minimum cut in an unweighted, undirected graph with n vertices and m edges. The fastest min-cut algorithms known today (for undirected, unweighted graphs) are:

- Deterministic: $O(mn)$ [11], [12].
- Randomized: $O(mlog^3n)$ [6].

The deterministic worst case running time for the algorithm is therefore $O(mn^2)$ (since the number of clusters is at most n). However, we usually expect $N << n$.

4.6.3 Properties of the HCS clustering

Theorem 4.1 *The diameter of each cluster at worst 2. That is, the distance between two vertices belonging to the same cluster is at most 2.*

Proof: Consider the graph $G(V, E)$ in the HCS iteration that found the cluster V. Let $k(G)$ be the size of the minimum cut, let $d(v)$ denote the degree of v and $\delta(G) = \min_v d(v)$. Observe that $k(G) \leq \delta(G)$: suppose on the other hand, that $d(v) < k(G)$ for some v, then the cut $(\{v\}, V \setminus \{v\})$ contradicts minimality of $k(G)$. Furthermore, if G is a cluster reported by the HCS algorithm, then according to the "if" statement in the algorithm $\frac{|V|}{2} \leq k(G)$ (otherwise G would have been divided in two by HCS). Therefore, $\delta(G) > \frac{|V|}{2}$.

Consider two vertices v_1 and v_2 in G. If they are neighbors, then surely the theorem holds for them. Let us therefore assume that they are not neighbors. From the previous inequality, each one of these vertices has more than $\frac{|V|}{2}$ neighbors in G. Therefore, they must have a common neighbor, since the total number of vertices in the graph is $|V|$. ■

While we have proven that each highly connected cluster has a small diameter, the converse does not necessarily hold. That is, G may have a subgraph with diameter 2 that is not a highly connected component.

Lemma 4.2 *Let S be a set of edges forming a minimum cut in the graph $G = (V, E)$. Let H and $\bar{H}$ be the induced subgraphs obtained by removing S from G, where $|V(\bar{H})| \leq |V(H)|$. If $|V(\bar{H})| > 1$ then $|S| \leq |V(\bar{H})|$, with equality only if $\bar{H}$ is a clique.*

Proof: Let $deg_s(x)$ denote the degree of vertex x in subgraph s. Therefore :

$$deg_s(x) + deg_{\bar{H}}(x) \geq |S|$$

Otherwise we can find a smaller cut with x as one side of the cut.
Summarize over all vertices in $\bar{H}$:

$$\sum_{x \in \bar{H}} deg_s(x) + \sum_{x \in \bar{H}} deg_{\bar{H}}(x) \geq |S||V(\bar{H})|$$

$\sum_{x \in \bar{H}} deg_s(x) \geq |S|$,because we do not count any edge in S more than once. (since vertices in $\bar{H}$ are not connected to each other in S).
$\sum_{x \in \bar{H}} deg_{\bar{H}}(x) \geq 2|E(\bar{H})|$, because we cannot count any edge in $\bar{H}$ more than twice (once for each vertex appearing in the edge). So :

$$|S| + 2|E(\bar{H})| \geq |S||V(\bar{H})|$$

Hence,

$$2|E(\bar{H})| \geq |S||V(\bar{H}) - 1|$$

If $\bar{H}$ is a clique,

$$|E(\bar{H})| = \frac{|V(\bar{H})||V(\bar{H}) - 1|}{2}$$

Therefore,

$$|S| \leq \frac{2|E(\bar{H})|}{|V(\bar{H}) - 1|} \leq \frac{|V(\bar{H}) - 1||V(\bar{H})|}{|V(\bar{H}) - 1|} = |V(\bar{H})|$$

■

Theorem 4.3 *Let S be a min cut in $G = (V, E)$, where $|S| \leq \frac{|V|}{2}$. Let H , $\bar{H}$ be the induced subgraphs obtained by removing S, where $|V(\bar{H})| \leq |V(H)|$. If $diam(G) \leq 2$ then:*

- *Every vertex in $\bar{H}$ is incident on S*
- *$\bar{H}$ is a clique and if $|V(\bar{H})| > 1$ then $|V(\bar{H})| = |S|$.*
  (Note : This is highly unlikely in clusters with random noise).

Proof:

- *case 1*: $|S| = \frac{|V|}{2}$.
  If $|V(\bar{H})| = 1$, it is trivial.
  If $|V(\bar{H})| > 1$, by Lemma 4.2, H, $\bar{H}$ are cliques with $\frac{|V|}{2}$ vertices. If there is $x \in \bar{H}$ that is not incident on S then $deg(x) < \frac{|V|}{2}$ and $(x, V \backslash x)$ is a smaller cut than S which is the min cut, a contradiction.

- *case 2*: $|S| < \frac{|V|}{2}$.
  Claim: $|V(\bar{H})| < |V(H)|$.
  Suppose $|V(H)| = |V(\bar{H})| = \frac{|V|}{2}$. Since $|S| < |V(H)| = |V(\bar{H})|$, then there is a path in G of length at least 3. Simply take two vertices ,one in H, and the other in $\bar{H}$, which are not in S.
  Therefore, $|V(H)| > \frac{|V|}{2} > |S|$. So $\exists v \in H$ which is not incident on S. Hence, every vertex in $\bar{H}$ must be incident on S (otherwise, like before, there is a path in G of length at least 3.).
  So $|S| \geq |V(\bar{H})|$. By Lemma 4.2, if $|V(\bar{H})| > 1$, $|V(\bar{H})| \geq |S|$. From the 2 inequalities we conclude $|S| = |V(\bar{H})|$, so by Lemma 4.2, $\bar{H}$ is a clique.

■

It can be shown, using this theorem, that the union of two vertex sets split by any step of HCS is unlikely to induce a graph with diameter ≤ 2 if noise is random, and the vertex sets are not too small. Another property of the solution is given by:

Theorem 4.4 *1. The number of edges in a highly connected subgraph is quadratic.*
2. The number of edges removed by each iteration of the HCS algorithm is at most linear. (but no guarantee that the overall number is small).

Proof: Let n (m) be the number of vertices (edges) in the graph. Then:

1. As we have seen before, $\frac{n}{2} < k(G) \leq \delta(G)$. Since the rank of each vertex is $> \frac{n}{2}$, the total number of edges is

$$N = \frac{1}{2}\sum_{v} \delta(v) > \frac{1}{2}\sum_{i=1}^{n} \frac{n}{2} = \frac{n^2}{4}$$

2. The algorithm removes the edges forming the minimal cut S, only if $|S| < \frac{n}{2}$. Therefore, obviously the number of removed edges is linear. ■

4.6.4 Refinements of HCS clustering

The algorithm as was introduced can be refined using the following heuristic methods:

```
Procedure HCS-LOOP($G(V,E)$)
  for ($i = 1$ to $p$) do
    $H \leftarrow G$
    repeatedly remove all vertices of degree $< d_i$ from $H$
    Repeat
      HCS($H$)
      perform orphan adoption
      remove clustered vertices from $H$
    Until (no new cluster is found)
    remove clustered vertices from $G$
end
```

Figure 4.9: Refinements of the HCS algorithm. $d_1, d_2, \ldots, d_p$ is a decreasing sequence of integers given as external input to the algorithm.

Singleton Adoption

The basic HCS algorithm may leave certain vertices as unclustered singletons. For that reason, each singleton is checked whether it fits in to one of the clusters. For each singleton x we compute the number of neighbors it has in each cluster and in the singleton set S. If the maximum number of neighbors is sufficiently large, and is obtained by a cluster (and not by S), then x is added to that cluster. The process is repeated a fixed number of times in order to accommodate the changes in clusters as a result of previous adoptions. This improvement is called *singleton adoption* or *orphans adoption.*

The low degree heuristic

When the input graph contains low degree vertices, one iteration of a minimum cut algorithm may simply separate a low degree vertex from the rest of the graph. This is computationally very expensive, not informative in terms of clustering, and may happen many times if the graph is large and sparse. Removing low degree vertices from the original graph before running the HCS algorithm eliminates such iterations and significantly reduces the running time. The complete algorithm, after refinements, is shown in Figure 4.9.

Cluster Merging

An additional refinement involves merging clusters that are similar enough into a single cluster. The rationale is that the algorithm tends to be too severe in its criteria for determining a cluster, and may split "true" clusters. Figure 4.10 shows the results of running the HCS algorithm with all the mentioned heuristics.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T11	T12	T13	T14	T15	T16	T17	T18	Total
S			7	6	2	9	19	21	9	1	7	5	43	14	9	10	21	23	206
C 1	1	1		6	2	5	13	18	5	2	17	6	16	7	25	9	51	85	269
C 2														**162**					162
C 3													**62**						62
C 4					1	15	1						4			1	1	**563**	587
C 5					5	1	2				4		2		2		**199**	2	217
C 6										**83**			2						85
C 7		1	1	1									2			**224**	2	1	232
C 8												**97**							97
C 9									**42**				1						43
C 10															**170**				170
C 11											**61**		1						62
C 12				1		2	4	4	5		2		7	4	7	4	10	31	81
C 13									**6**										6
C 14													1			**26**			27
C 15			4										5			4		3	16
C 16																**6**			6
Total	1	2	12	14	10	32	39	43	67	86	91	108	146	187	213	284	285	708	2329

Figure 4.10: Results of HCS with Cluster merging: T: True clusters; C: HCS+merge. (i, j): number of common elements in $C(i)$ and $T(j)$. 13 out of 16 clusters are *almost pure*, 6 are *completely pure*.

4.6.5 Assessing Clustering Quality

We now describe measures for the quality of a solution given a true clustering. Let T be the "true" solution and S the solution we wish to measure. Denote by n_{11} the number of pairs of elements that are in the same cluster in both S and T. Denote by n_{01} the number of pairs that are in the same cluster only in S, and by n_{10} the number of pairs that are in the same cluster only in T. We define the *Minkowski Score* to be:

$$D_M(T,S) = \sqrt{\frac{n_{01}+n_{10}}{n_{11}+n_{10}}}$$

In this case the optimum score is 0, with lower scores corresponding to better solutions.

An alternative is the *Jaccard Score*:

$$D_J(T,S) = \frac{n_{11}}{n_{11}+n_{10}+n_{01}}$$

Here the optimum score is 1, with larger scores corresponding to better solutions.

4.6.6 Simulation Results

Intensive tests of the HCS algorithm on simulated data were performed. The simulation process computes artifical gene fingerprints (hybridized oligos) for each participating gene. For each gene and a given probe, the precise locations along the gene are generated in a realistic manner. Then, truncated clones of each gene are generated. Each clone inherits the probe fingerprints and their locations from its original gene (just the fingerprints with locations within the clone boundaries are inherited). Finally, each copy is incorporated with false positive and false negative errors, again, realistically. If we denote the total number of oligos by p and the total number of clones by N, then the result of the simulation is an $N \times p$ hybridization matrix H, where $H_{ij} = 1$ if clone i hybridized with oligo j, and $H_{ij} = 0$ otherwise.

The simulation results are summarized in Figure 4.11. A comparison to the Greedy algorithm is given in Figure 4.12.

4.6.7 Clustering Real cDNA Data

In addition, a test of clustering real cDNA data was performed. (see [5]). The input contained 2329 cDNAs, originating from 18 genes. The true clustering, obtained by hybridization with long, unique sequences, is given in Table 4.1.

The high variability in abundance of genes can be easily seen. The results of the test are summarized in Figure 4.13.

Cluster	Cluster size	Gene name
T_{18}	709	Ef1 alpha
T_{17}	285	clone 190B1
T_{16}	284	Cytochr c oxi
T_{15}	213	tubulin beta
T_{14}	187	40SRibo protS6
T_{13}	146	40SRibo protS3
T_{12}	108	40SRibo protS4
T_{11}	91	GAPDH
T_{10}	86	60SRibo protL4
T_9	67	Ef1 beta
T_8	43	Human calmodulin
T_7	39	heat shock cogKD71
T_6	32	heat shock cogKD90
T_5	14	Human TNF recep
T_4	12	Human AEBP1
T_3	10	clone 244D14
T_2	2	clone 241F17
T_1	1	Human anion ch

Table 4.1: True clusters in a cDNA fingerprinting experiment. Results are shown in Figure 4.13.

In 14 out of the 17 clusters generated by the algorithm, over 92% of the entities belong to the same gene (true cluster). Those clusters are called *almost pure*.

As the correct clusters are not known (in real experiments), and the main goal of cDNA clustering is to avoid repeated sequencing of cDNAs originating from the same gene, the following strategy can be used: From each cluster, up to 10 cDNAs are randomly chosen to be sequenced. If, for example, 9 out of 10 give the same sequence, the cluster is with high certainty almost pure, and no more sequencing of its members is needed. Otherwise, all the members of the cluster are sequenced. This strategy could have saved about 75% of the sequencing cost in this dataset.

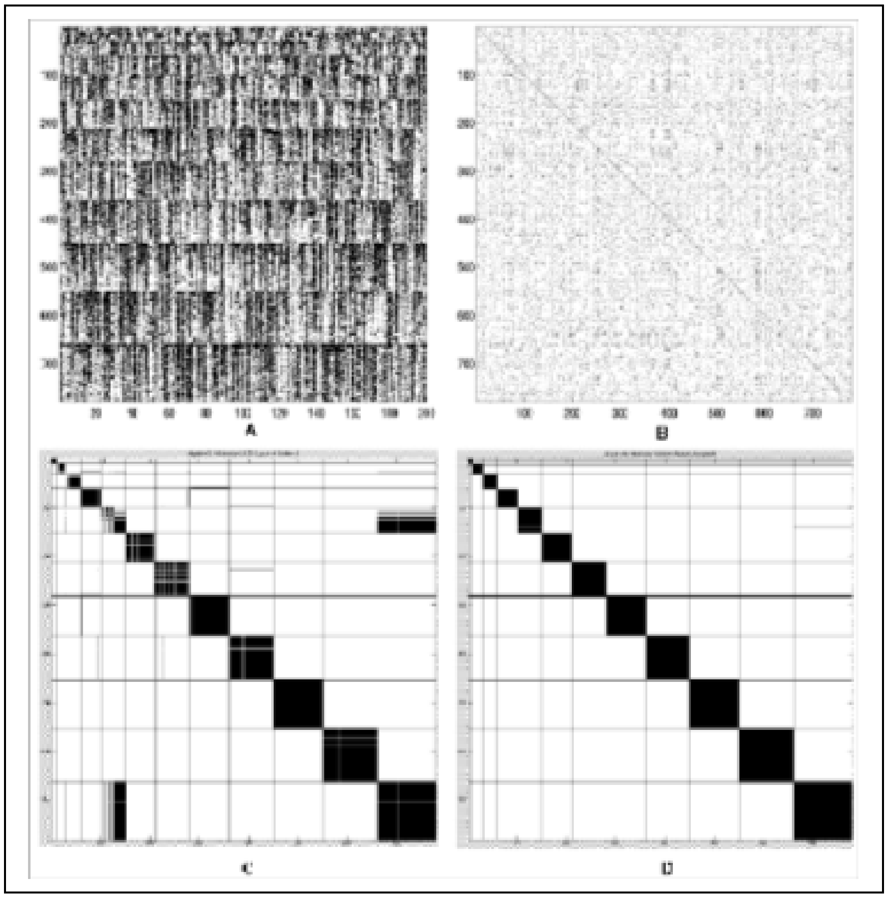

Figure 4.11: Examples of results of HCS and Greedy clustering algorithms in high noise simulation. The fingerprint data consisted of 780 cDNAs from 12 genes, in clusters of sizes 10,20,...,120. The number of oligos is 200. The expected rate of false positive hybridizations is 25%. The expected false negative hybridization rate is 40%. A: The hybridization fingerprints matrix H. Each of the 780 rows is a fingerprint vector of one cDNA. White denotes positive hybridization. B: The binary similarity matrix. Position i, j is black iff $S_{ij} > 50$. Matrix coordinates are scrambled, as in realistic scenarios. C: Clustering solution generated by the greedy algorithm. Minkowski score is 1.32. cDNAs from the same true cluster appear consecutively, and the black lines are the borders between the different clusters. Position i, j is black if the solution places cDNAs i and j in the same cluster. D: Clustering solution generated by the HCS algorithm. Minkowski score is 0.209.

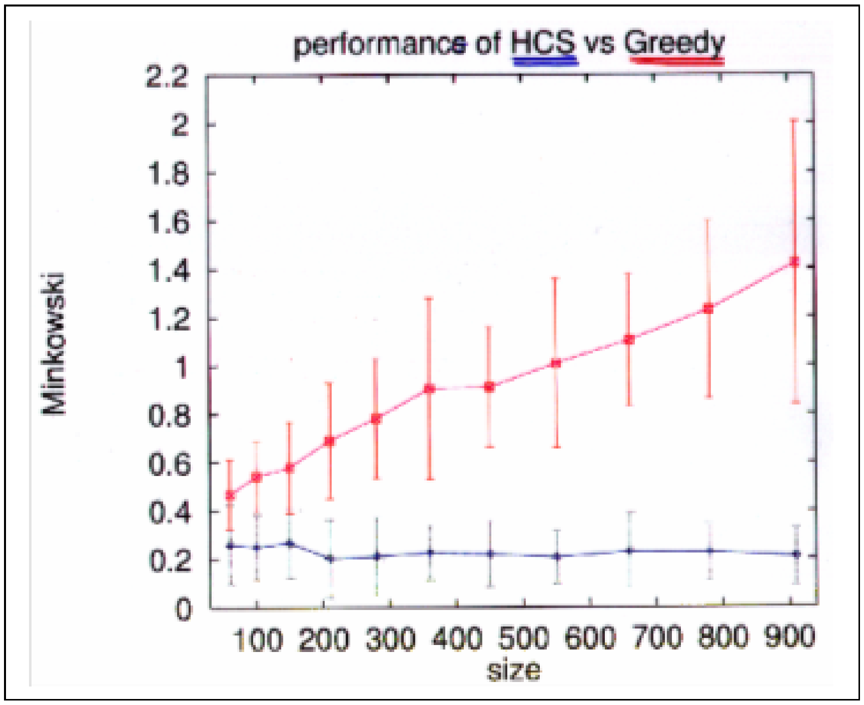

Figure 4.12: Performance comparison of HCS (squares) and Greedy (diamonds) algorithms on simulated data (using Minkowski score).

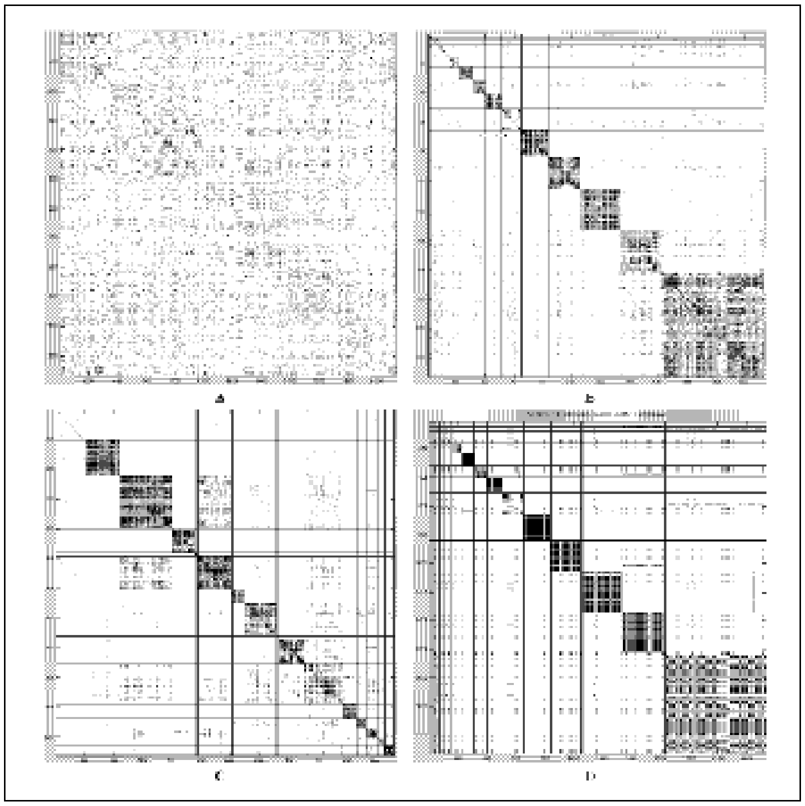

Figure 4.13: Clustering results on real cDNA data (true clusters are listed in Table 4.1). A: The binary similarity matrix. A block point appears at position (i, j) iff $S_{ij} \geq 110$. B: Reordering of A according to the true clustering. cDNAs from the same true cluster appear consecutively, and the black lines are the borders between the different clusters. C: Reordering of A according to the clustering produced by the HCS algorithm. Clusters appear in the order of detection. D: Comparison of the algorithmic solution and the true solution. Rows and columns are ordered as in B. Position (i, j) is black iff the algorithm places i and j in the same cluster.

Bibliography

[1] W. Bains and G. C. Smith. A novel method for nucleic acid sequence determination. *J. Theor. Biology*, 135:303–307, 1988.

[2] R. Drmanac and R. Crkvenjakov, 1987. Yugoslav Patent Application 570.

[3] 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.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.

[5] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, and R. Shamir. An algorithm for clustering cDNA fingerprints. *Genomics*, 66(3):249–256, 2000. A preliminary version appeared in Proc. RECOMB '99, pp. 188–197.

[6] D.R. Karger. Minimum cuts in near linear time. In *Proc. 28th annual ACM symposium on theory of computing*, pages 56–63, 1996.

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

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

[9] S. C. Macevics, 1989. International Patent Application PS US89 04741.

[10] 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.

[11] D.W. Matula. Determining edge connectivity in $O(nm)$. In *Proceedings 28th IEEE Symposium on Foundations of Computer Science*, pages 249–251, 1987.

[12] H. Nagamochi and T. Ibaraki. Computing edge connectivity in multigraphs and capacitated graphs. *SIAM J. Disc. Math.*, 5:54–66, 1992.

[13] 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.

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

[15] E. Southern, 1988. UK Patent Application GB8810400.

[16] 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.