Algorithms for Molecular Biology **Fall Semester, 2000**

Lecture 12: January 15, 2001

Lecturer: Ron Shamir *Scribe: Shachar Ofek and Elena Zotenko*[1]

12.1 Clustering Gene Expression Data

12.1.1 Overview and Motivation

In any living cell that undergoes biological process different subsets of its genes are expressed in different stages of the process. The particular genes expressed at a given stage and their relative abundance are crucial to the cell's proper function. Analysis of gene expression patterns can provide an insight into gene/function relationships, effects of experimental treatments for diseases, and many other molecular biological processes.

Clustering techniques applied to gene expression data will partition genes into groups/clusters based on their expression patterns. Genes in the same cluster will have similar expression patterns, while genes in different clusters will have distinct well-separated expression patterns. We will see several examples of gene expression clustering in section 12.1.4.

Gene expression data can be represented by a real-valued *expression matrix* I, where $I_{i,j}$ is the measured expression level of gene i in experiment j. Experiments can be different time points, different body tissues or different strains of the organism. The i-th row of the expression matrix is called the *expression pattern* or the *transcriptor* for gene i.

The similarity between expression patterns of any two genes is represented by a *similarity matrix* S, where $S_{i,j}$ is the similarity level between the expression patterns of gene i and gene j. The similarity matrix can be derived from the expression matrix by applying some similarity measure to expression patterns of every pair of genes. Examples of similarity measure are Euclidean distance and Pearson correlation.

The similarity matrix can be further transformed into *similarity graph* G_θ, whose vertices are genes and $(i,j) \in E(G_\theta)$ iff $S_{i,j} > \theta$.

12.1.2 The Model of the Corrupted Clique Graph

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

[1] this scribe is partially based on scribe by Dror Irony and David Oren

H (vertices are genes and cliques are clusters). Contamination errors introduced into gene expression data result in similarity graph a $C(H)$ 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

This model for the clustering problem can be reduced to a clique graph editing problem, stated as follows.

Problem 12.1 *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 [5].

Problem 12.2 *clique graph completion problem*
INPUT: $G(V, E)$ a graph.
OUTPUT: $Q(V, F)$ a clique graph with $E \subseteq F$ which minimizes $|E \Delta F|$.

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

The clique graph deletion problem is NP-hard [3]. Moreover, any constant factor approximation to the clique graph deletion problem is NP-hard as well [5].

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 w.h.p. (with high probability).

One of the simplest probabilistic models for contamination errors is a *random corrupted clique graph*. The contamination errors are represented by flipping a fraction $p < 0.5$ of the edges of the original clique graph H. This can be thought of as flipping edges with probability $p < 0.5$. 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.

12.1.3 The BioCluct Clustering Algorithm

In this section we present an example of a clustering algorithm, BioClust. In subsequent sections we will see the results of applying BioClust to synthetic data and to real biological data. Material described here is from [1].

We will say that a clique graph $H(V, E)$ has a *γ-cluster structure*, if the size of each clique in H is at least $\gamma|V|$.

For a fixed $0 < \gamma < 1$ we say that a randomized algorithm A *reconstructs γ-cluster structures w.h.p.* 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 clique graph with n vertices and γ structure, $Prob(A(G) \neq H) < \delta$.

First we describe a theoretic algorithm which reconstructs γ-cluster structures w.h.p. The correctness proof of the algorithm will be outlined. We then present the practical heuristic, which is based on the theoretical algorithm and preserves most of its properties.

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 S', by putting each vertex $v \in S'$ into the cluster $C = argmax_i(\frac{|\{u|u \in C_i,(u,v) \in E\}|}{|C_i|})$.

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

Algorithm Outline

Suppose we are given $G(V, E)$ 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 $m = \lceil \frac{1}{\gamma} \rceil$.

The algorithm will perform the following steps (please refer to figure 12.1):

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 a clustering $\{C_1^1, ..., C_m^1\}$ of $S_0 \cup S_1$;
   (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 $|E \Delta E(C)|$;

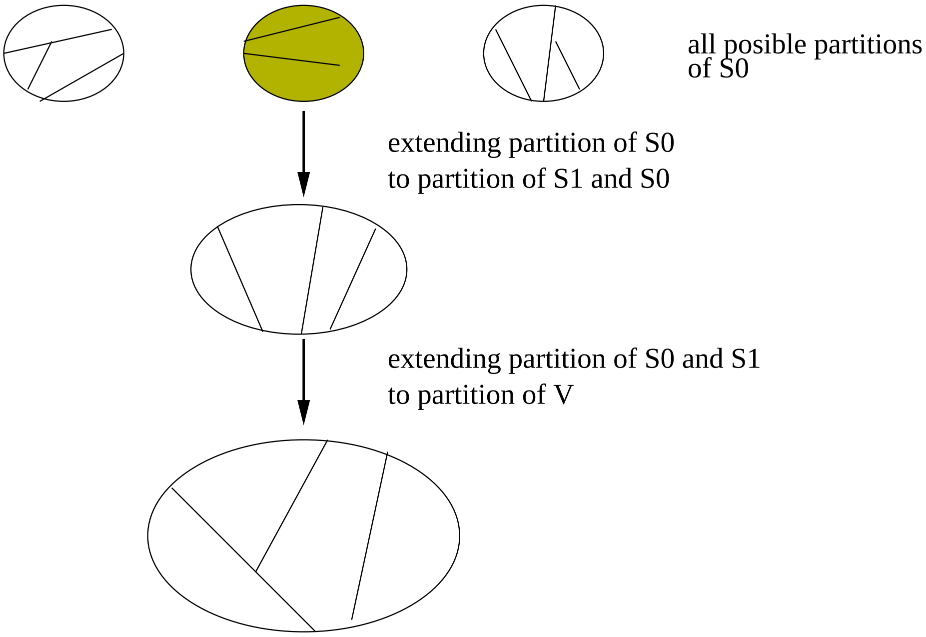

Figure 12.1: Algorithm steps are shown schematically.

Algorithm 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 $m^{|S_0|} = m^{\log\log(n)} = \log^c(n)$;

- For each partition we perform $O(\log(n)\log\log(n))$ operations in step **3.a** and $O(n\log(n))$ operations in step **3.b**, 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 [1].

Theorem 12.1 *Chernoff 1952 [2]*
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 a relative entropy between* $(p, 1-p)$ *and* $(a, 1-a)$ *and is defined by* $D(a\|p) = p\log(\frac{p}{a}) + (1-p)\log(\frac{1-p}{1-a})$.

Theorem 12.2 *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 a simple 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 clustering which is H. 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 12.3 *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 following holds w.h.p.* $|E(H)\Delta E(G)| < |E(C)\Delta E(G)|$.

Proof (outline)

First we will show that given some 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 12.2 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 $m = \lceil \frac{1}{\gamma} \rceil$. 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 $|E(C)\Delta E(H)| < \frac{n\log(n)}{D(0.5\|p)}$, similar but more complicated argument shows that w.h.p. for any such C $|E(H)\Delta E(G)| < |E(C)\Delta E(G)|$;

$\blacksquare$

Practical Implementation

Based on ideas of theoretical algorithm given in previous sections, a simple and practical heuristic was implemented. All the tests described in subsequent sections were performed using this practical implementation of the theoretic 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 *affinity threshold* τ we will say that v is *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 move is possible;
- Repeat until all genes are clustered;

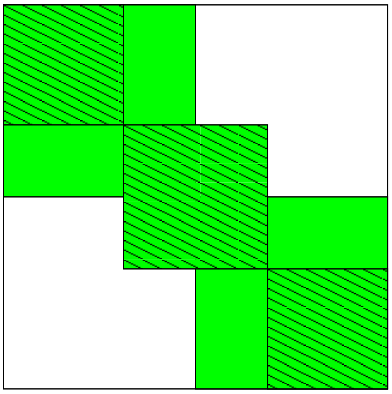

Figure 12.2: Schematic proof of the key observation. This figure depicts a graph G (blank), original clique graph H (green) and some other clique graph C (green with pattern). Consider $E(H)\Delta E(G)$ and $E(C)\Delta E(G)$. Each edge (u,v) which belongs to neither C nor to H, but belongs to G increments 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 $E(C)\Delta E(H)$. But $|E(C)\Delta^* E(G)| + |E(H)\Delta^* E(G)| = |E(H)\Delta E(C)|$. 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 $|E(H)\Delta^* E(G)| > \frac{|E(H)\Delta E(C)|}{2}$.

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 allows also to remove a weak element from a cluster;

Although nothing 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.

cluster structure	n	p	matching coeff.	Jaccard coeff.
$\{0.4, 0.2, 0.1, 0.1, 0.1, 0.1\}$	500	0.2	1.0	1.0
$\{0.4, 0.2, 0.1, 0.1, 0.1, 0.1\}$	500	0.3	0.999	0.995
$\{0.4, 0.2, 0.1, 0.1, 0.1, 0.1\}$	500	0.4	0.939	0.775

Table 12.1: Performance of the BioClust algorithm for different values of p. Mean values of matching coefficient and Jaccard coefficient are given.

12.1.4 Clustering Using BioClust

Synthetic Data

BioClust is tested on the members of $\Omega(H, p)$. The simulation procedure is as follows (please refer to figure 12.3 for visualization of the simulation procedure):

- 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 clique graph H;

There are several comparison criteria, which can be used to compare the algorithm's output to the original clique graph. 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.

Table 12.1 presents results of simulation for different values of contamination error p. 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 on the performance of the algorithm.

Figure 12.4 presents results of simulation 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 increases.

Temporal Gene Expression Data

The gene expression data used in this experiment is from [8]. In this paper the authors study the relationship among expression patterns of genes involved in the rat Central Nervous

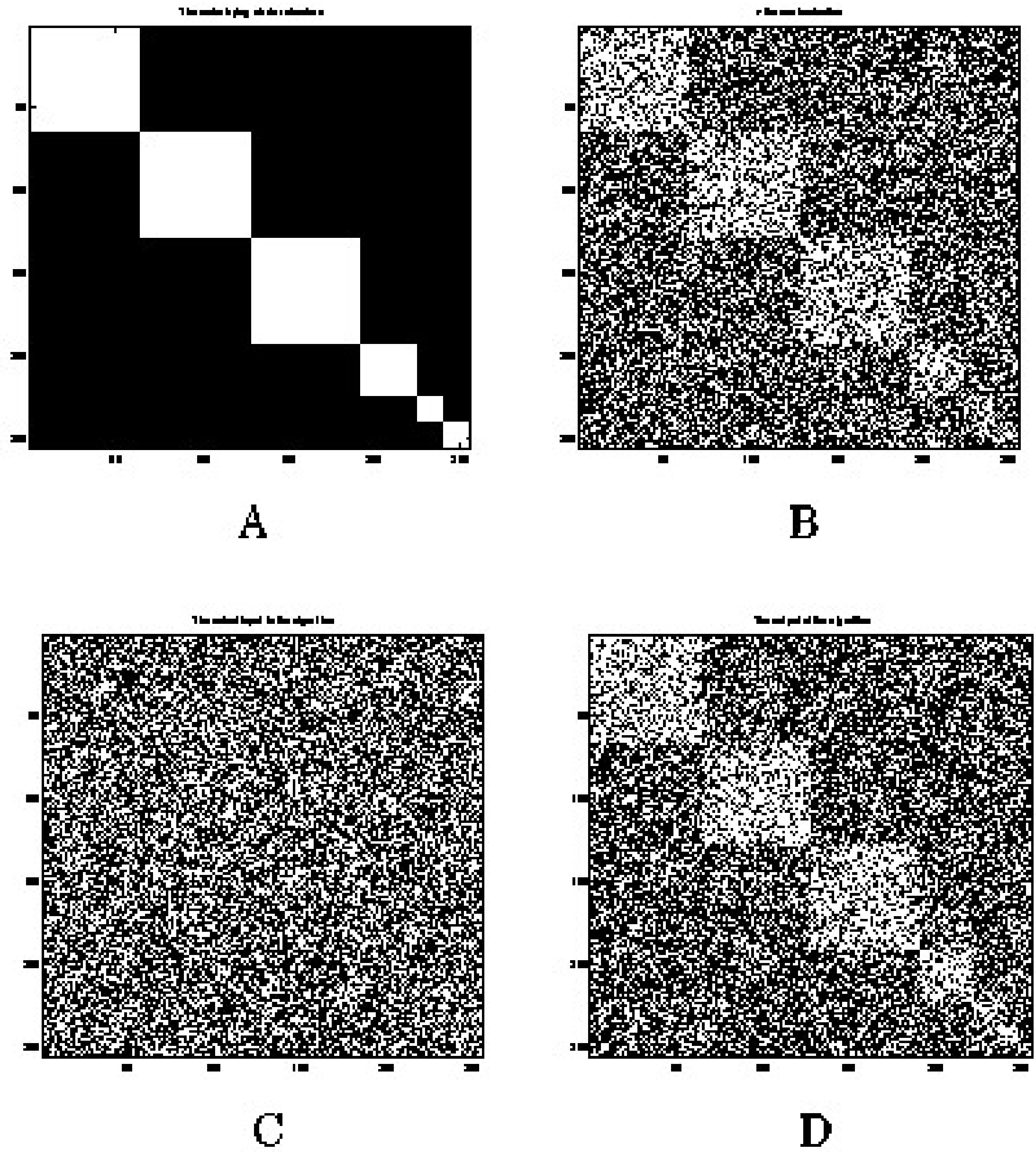

Figure 12.3: A visualization of the simulation procedure. **A**: The adjacency matrix of the original clique graph H before introduction of errors. Position (i,j) is white if $(i,j) \in E(H)$, that is, if i and j belong to the same cluster. **B**: The same matrix after introduction of errors. Note that 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.

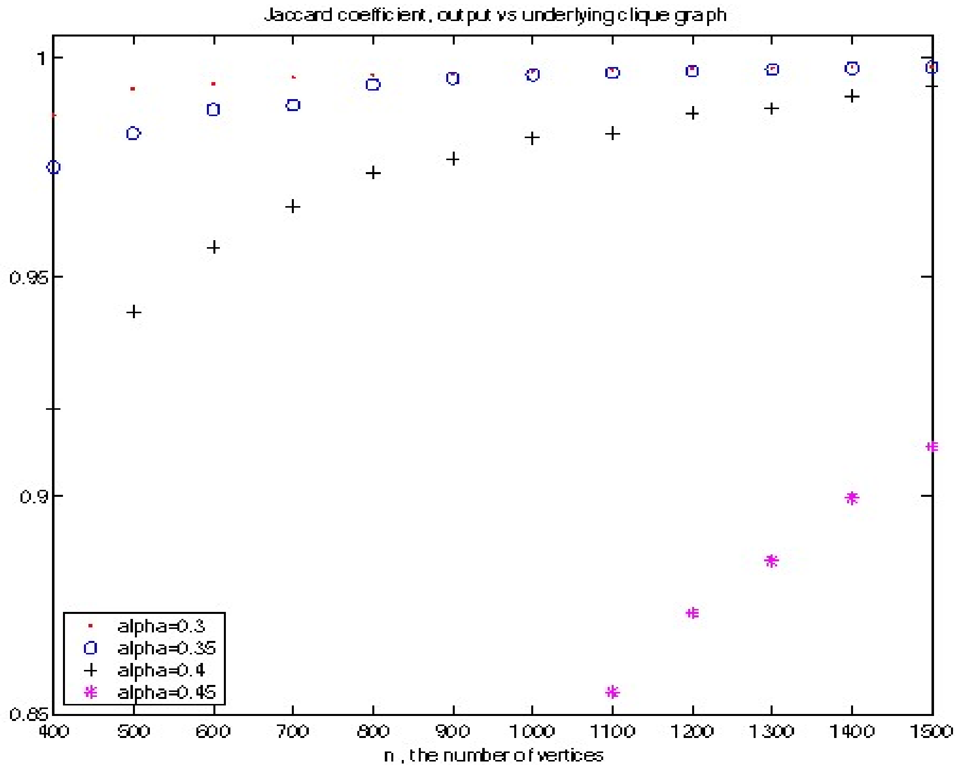

Figure 12.4: Simulation results for H with cluster structure of $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}\}$. The x-axis is the number of clustered entities n and y-axis is the mean value of the Jaccard coefficient. Each curve corresponds a specific probability α of contamination error.

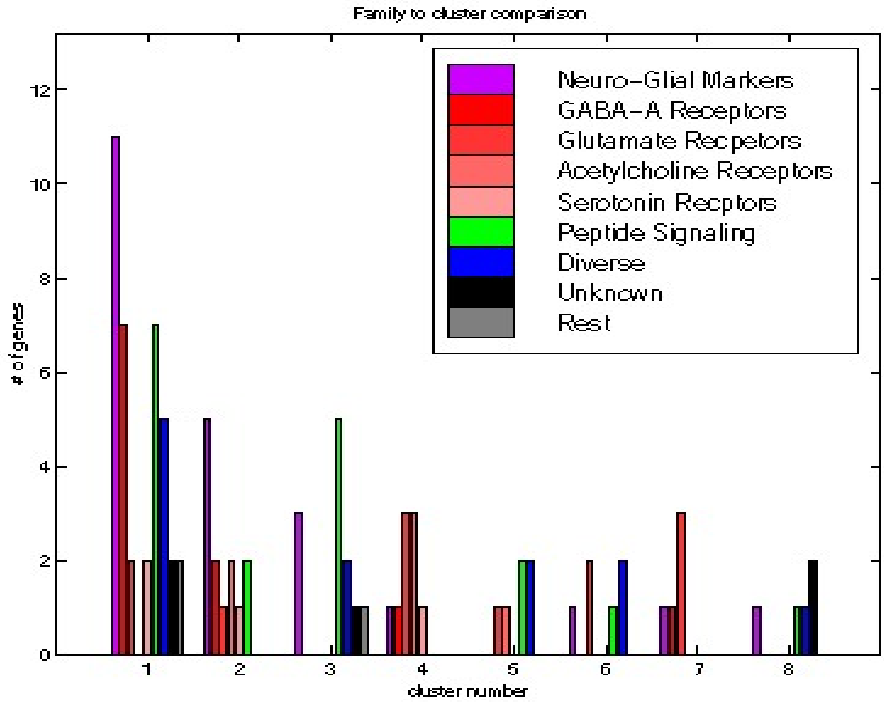

Figure 12.5: Applying the algorithm to temporal gene expression data [8]. The solution generated by the algorithm is compared to the prior classification. For each cluster (x-axis), bars composition in terms of biologically defined families. The height of each bar (y-axis) represents the number of genes of a specific cluster family. Most clusters contain predominantly genes from one or two families.

System (CNS).

Gene expression patterns were measured for 112 genes along nine different development time points. The experiments here are the 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 112×17 expression matrix. The similarity matrix was obtained using Euclidean distance.

Eight clusters were obtained. Since partitioning to clusters is known from [8] this experiment was done mainly for validation of algorithm.

Figures 12.5 and 12.6 present the clustering results. Note that all clusters, perhaps with exception of cluster #1, manifest clear and distinct expression patterns. Moreover, the agreement with the prior biological classification is quite good.

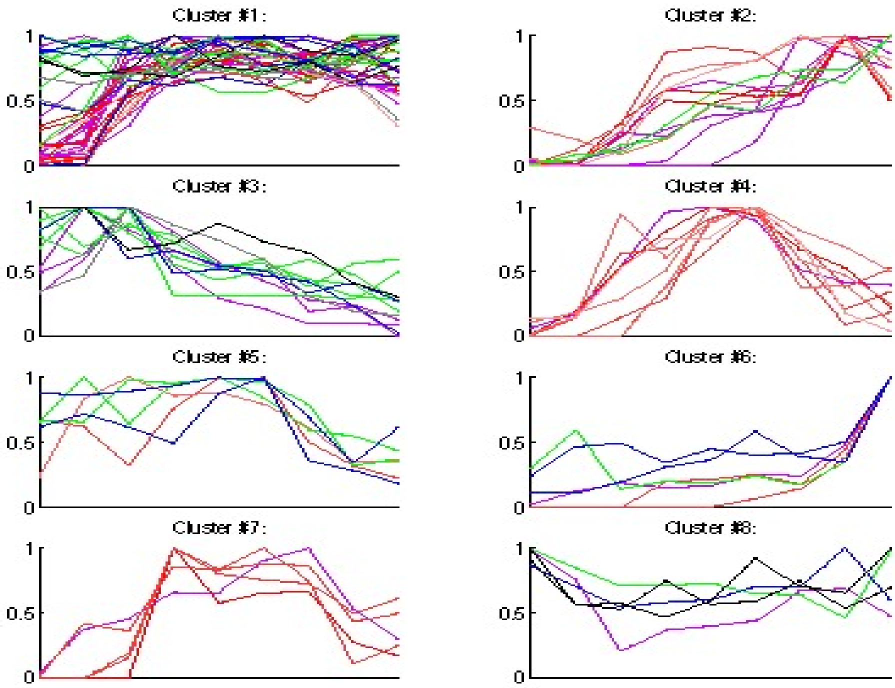

Figure 12.6: Applying the algorithm to temporal gene expression data [8]. Each graph presents expression patterns of genes in a specific cluster. The x-axis represents time, while the y-axis represents normalized expression level.

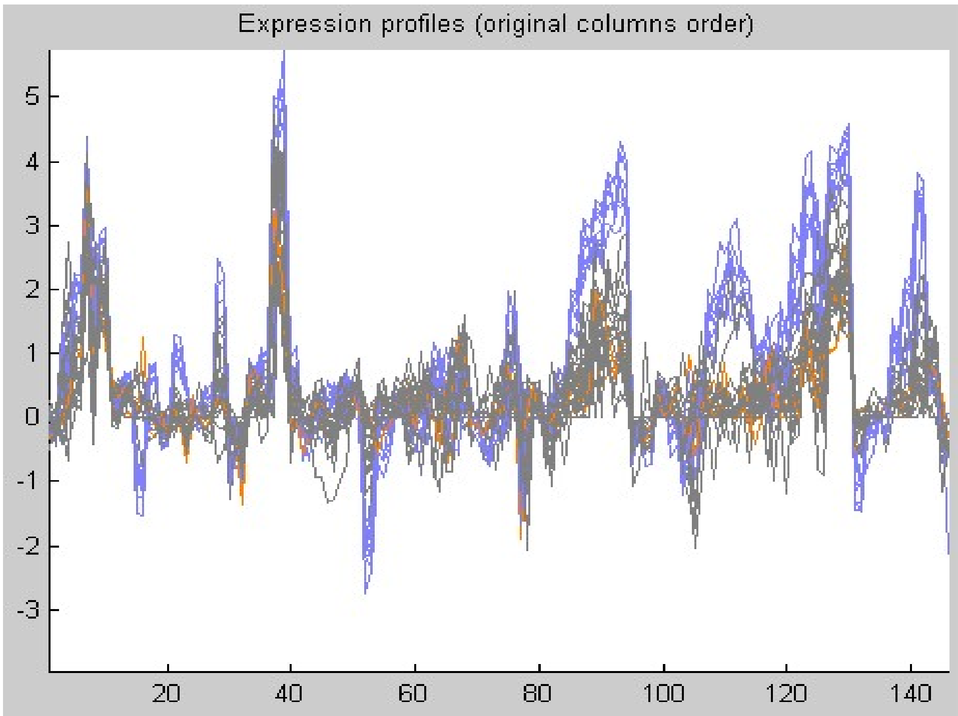

Figure 12.7: Expression patterns of cluster #1. This cluster contains all the known sperm proteins out of all the 1246 genes.

Multi-Condition Gene Expression Data

The gene expression data used in this analysis is from [6]. Kim et al. studied gene regulation mechanisms in C.elegans.

Gene expression patterns were measured for 1246 genes in 146 experiments, resulting in 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 families. The algorithm clustered these families quite well into few homogeneous clusters. Figure 12.7 shows the expression patterns #1 which contains all the known sperm proteins out of all the 1246 genes.

One example of the potential use of clustering for analyzing gene expression patterns is shown in figure 12.8. A six-gene cluster containing two growth-related genes and four anonymous genes. This suggests the possibility that the other four genes are also growth related.

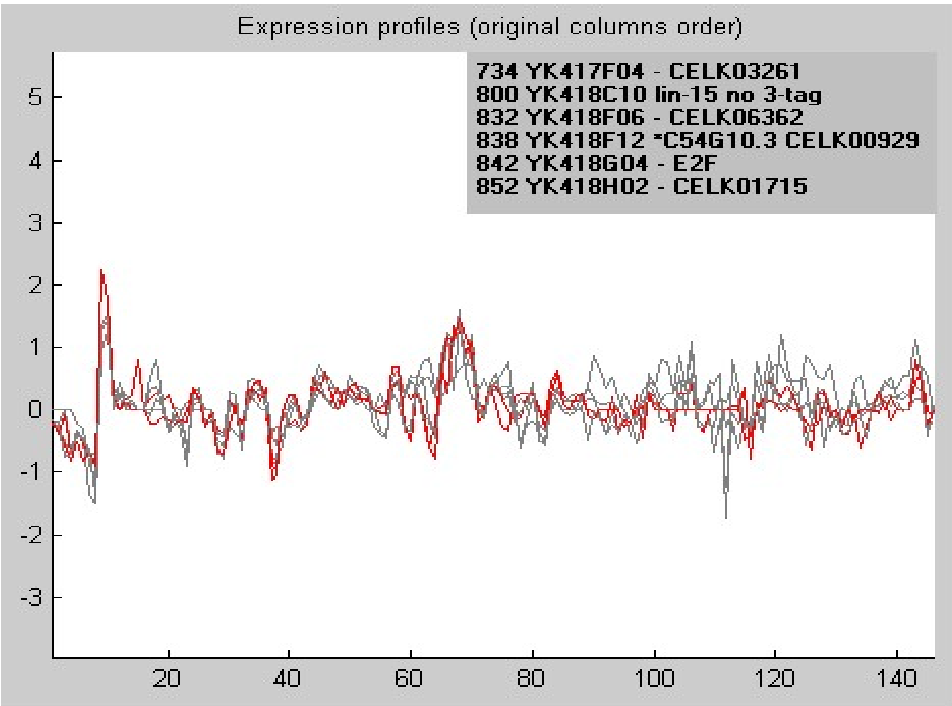

Figure 12.8: Expression patterns and members of cluster #24. This cluster contains two growth-related genes *lim-15* and *E2F*. This suggests the hypothesis that the other four members of this cluster have related function.

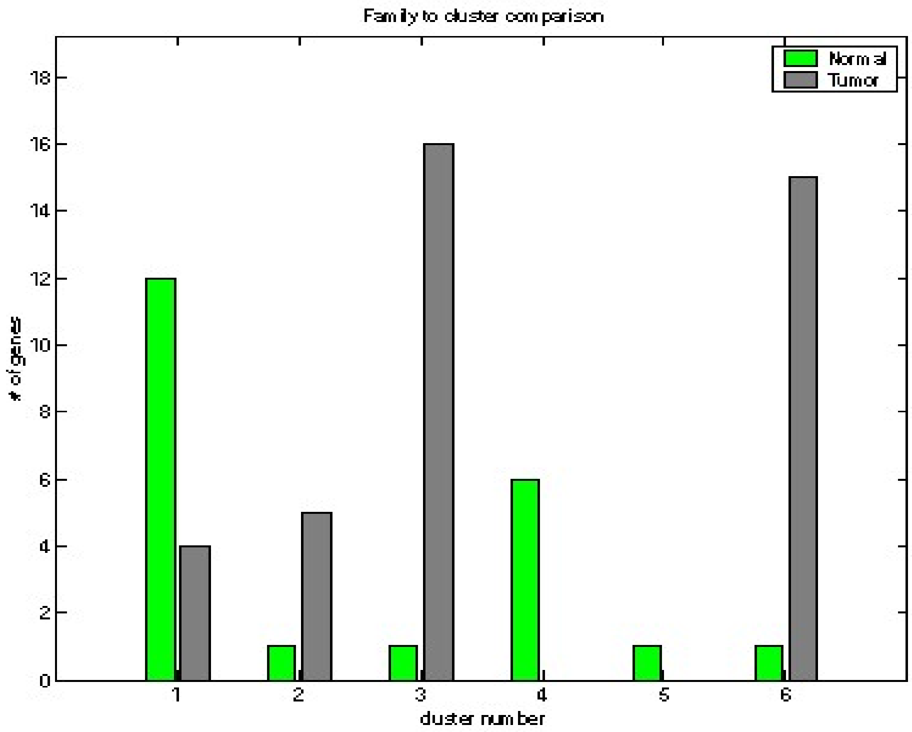

Figure 12.9: Distribution of tumor and normal tissues in the six clusters produced by the algorithm.

Tissue Gene Expression Data

The gene expression data used in this experiment is from [7]. The authors describe an analysis of gene expression data obtained form 40 tumor and 22 normal colon tissues.

Gene expression patterns were measured for 2000 genes in 62 tissues. The similarity was measured with Pearson correlation.

BioClust formed 6 clusters. Figure 12.9 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.

12.1.5 Sequencing by Hybridization

Standard oligo chips can, at least theoretically, be used for sequencing. Let us prepare an oligo chip that contains all possible sequences of length k. These sequences are called k-*mers*.

Practical values of k are 8-10. If we expose this chip to a solution containing some target DNA, the results will show which k-mers occur in the target sequence.

Definition The *k-spectrum* of sequence T is the multi-set of all its substrings of length k.

Note, that the *k-spectrum* is a *multi-set*. We assume that if a k-mer appears more than once, in the target DNA, the hybridization experiment will report the number of its occurrences. To date, this requirement is impractical.

Problem 12.4 Reconstructing a sequence from hybridization data
INPUT: A multi-set S of k-mers
QUESTION: Does exist a sequence T such that S is the k-spectrum T? If yes, find T.

For instance, for $k = 3$:

$$\begin{aligned} T &= ATGCAGGTCCAG \\ S &= \{ATG, AGG, CAG, GCA, GGT, GTC, TCC, TGC, CCA, CAG\} \end{aligned}$$

The naive approach

Let us define a directed graph $G' = (V', E')$, where $V' = \{\text{existing } k\text{-mers}\}$ and an edge $E' = (v_1, v_2)$ exists iff the last $k-1$ characters of v_1 match the first $k-1$ characters of v_2 (i.e., it is possible that v_2's index in T, is the successor of v_1's index.

The problem is now to find a Hamiltonian path in the directed graph G. However, it is a common knowledge that the Hamilonian path problem is NP-Complete. Therefore, this solution cannot be used for large input sets.

The polynomial solution

Luckily, a polynomial solution for this problem exists, due to Pevzner [4]. Define another directed graph $G = (V, E)$. This time the vertices will be $(k-1)$-mers. An edge $e = (v_1, v_2)$ is introduced if the first $k-2$ characters of v_1 match the last $k-2$ characters of v_2, and the concatenation of the first character of v_1, with the $k-2$ common characters and the last character of v_2 form a k-mer that was reported present in the sequence. This graph is called the *de-Bruijn* graph of the sequence.

For instance, if

$$S = \{AAA, AAC, ACA, CAC, CAA, ACG, CGC, GCA, ACT, CTT, TTA, TAA\} \quad (12.1)$$

then the corresponding graph will be the one in figure 12.10. The mathematical problem here is to find an *Euler path*, that is a path that uses each edge once, and only once. For example, the sequence

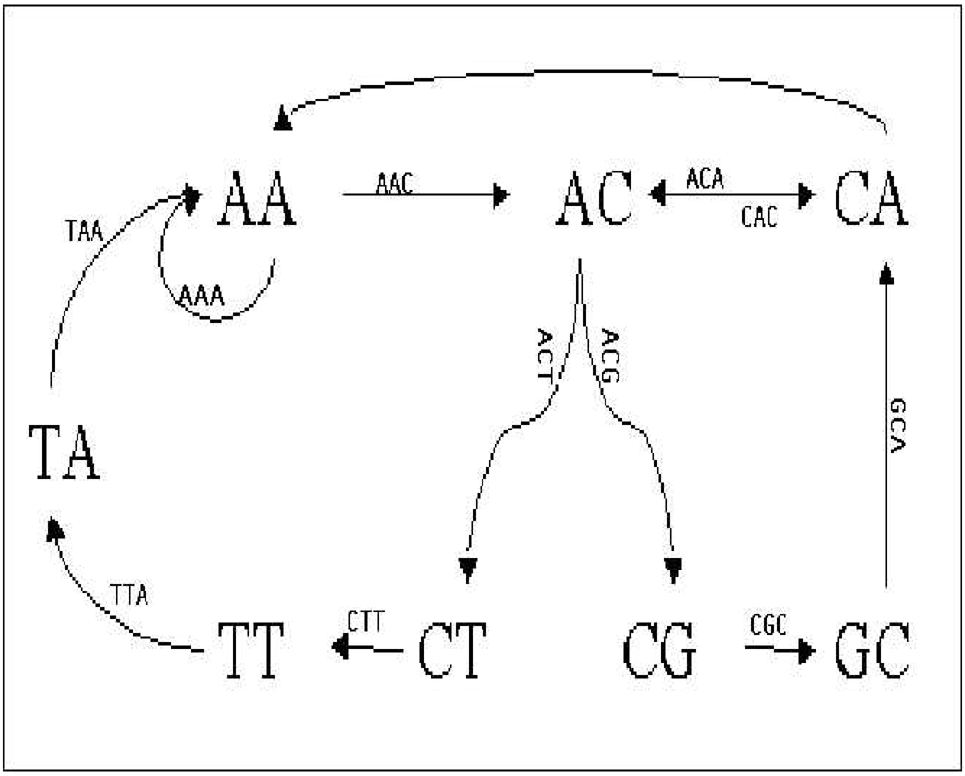

Figure 12.10: The $(k-1)$-mer graph constructed given the k-spectrum $(k=3)$ in equation 12.1.

$$T = ACAAACGCACTTAA$$

is a solution to the instance whose graph is depicted in figure 12.10, corresponding to the Euler path

$$AC \to CA \to AA \to AA \to AC \to CG \to GC \to CA \to AC \to CT \to TT \to TA \to AA$$

It should be noted that for this construction, it is *very important* to know whether a given k-mer occurs more than once in the target sequence. For instance, if ACA occurs two times in S, then there should be two edges between AC and CA. Otherwise, our solution will not be correct.

While this solution is mathematically elegant, there are several problems with using it in true biological context:

1. For some graph configurations, there is more than once Euler path. In such cases we will not be able to reconstruct the sequence. For an example of such a graph, see figure 12.10.

2. As in all biological experiments, the spectrum we measure contains a large proportion of errors. This solution is not robust enough to handle them.

3. A related problem is that of edge multiplicity. We can consider ourselves lucky to know with certainty whether a certain k-mer occurs in our sequence. In most cases we have no way of knowing exactly how many times it occurs.

The only problem which we shall address here is the first one – the problem of the graph representing more than one possible sequence. Figure 12.11 shows the probability that a random string S cannot be uniquely reconstructed from its k-spectrum.

Some other chip designs achieve a somewhat better result, but these designs are only theoretical. They are usually very difficult or impossible to manufacture, and cannot be used in true biological context.

Sadly, then, sequencing based on hybridization is not a real alternative to standard sequencing.

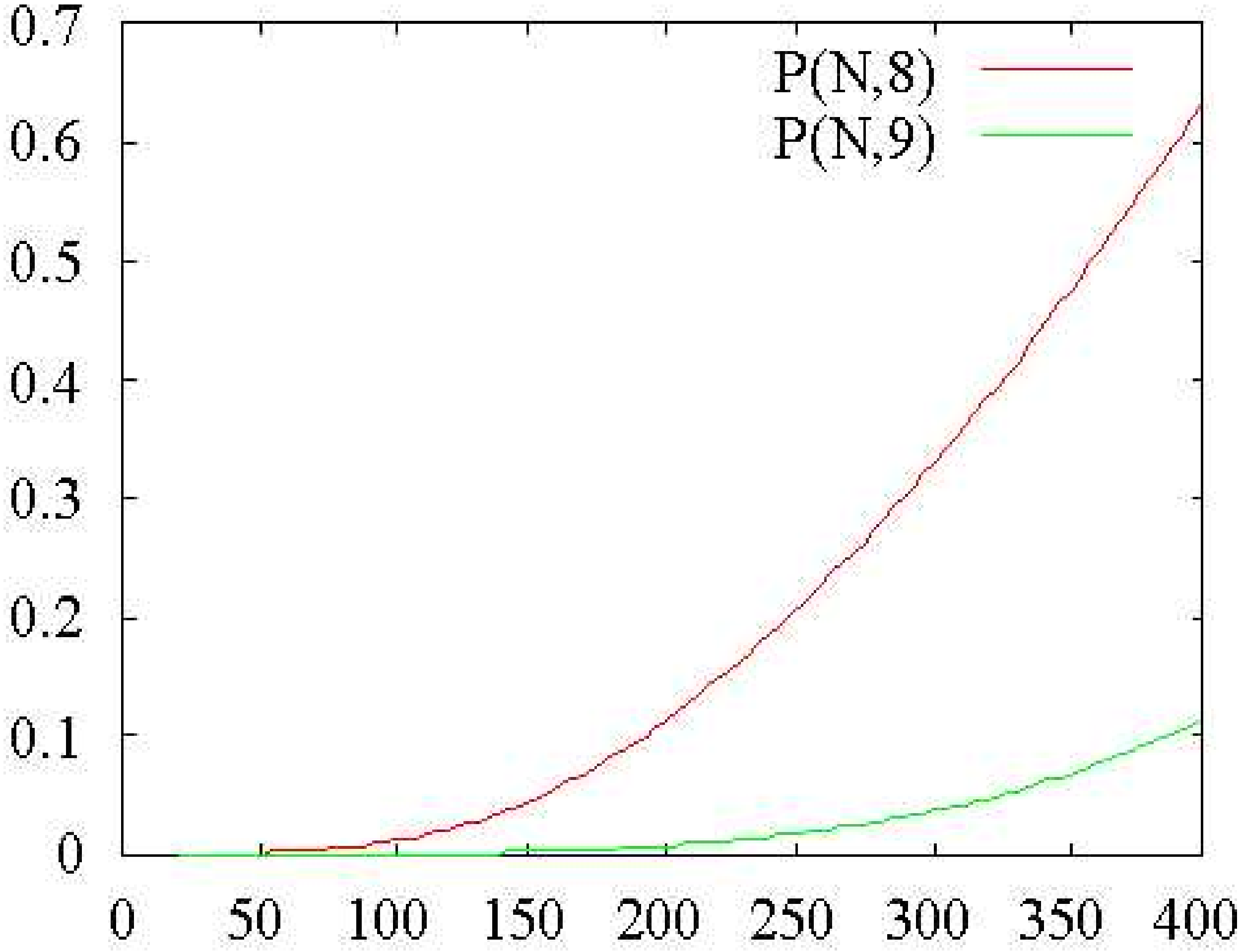

Figure 12.11: $P(N,k)$ is the probability that for a random string S of length N there exists a sequence S_0, whose k-spectrum equals to S's k-spectrum.

Bibliography

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

[2] H.Chernoff. A measure for the asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. of Math. Stat.*, 23:493 – 509, 1952.

[3] Assaf Natanzon, Ron Shamir, and Roded Sharan. A polynomial approximation algorithm for the minimum fill-in problem. *SIAM Journal on Computing*, 30(4):1067–1079, August 2001.

[4] P.A.Pevzner. l-tuple dna sequencing:computer analysis. *J. Biomol. Struct. Dyn.*, 7:63 – 73, 1989.

[5] Roded Sharan and Ron Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In Russ Altman, L. Bailey, Timothy, Philip Bourne, Michael Gribskov, Thomas Lengauer, and Ilya N. Shindyalov, editors, *Proceedings of the 8th International Conference on Intelligent Systems for Molecular (ISMB-00)*, pages 307–316, Menlo Park, CA, August 16–23 2000. AAAI Press.

[6] S.Kim and et. al. Department of Development Biology, Stanford University.

[7] U.Alon and et. al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS*, 96:6745 – 6750, 1999.

[8] X.Wen and et. al. Large-scale temoparl gene expression mapping of central nervous system development. *PNAS*, 95:334 – 339, 1998.