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

Lecture 9: June 13, 2002

Lecturer: Benny Chor and Ron Shamir *Scribe: Meital Levy and Giora Unger*[1]

9.1 The Order-Preserving-Sub-Matrix (OPSM) Problem

9.1.1 Introduction

In previous lectures we studied various clustering methods and algorithms. We then moved on, to talk about methodologies that can be termed "beyond clustering", such as biclustering and the Plaid model. We will complete this subject by presenting another approach to finding meaningful patterns in gene expression matrices - the Order Preserving Sub-Matrix (OPSM) method [4].

9.1.2 Motivation

Recall that standard clustering methods for pattern discovery in gene expression matrices are based on clustering genes by comparing their expression levels in all experiments, or clustering experiments by comparing their expression levels for all genes. OPSM goes beyond such global approaches by looking for local patterns that manifest themselves when we focus simultaneously on a subset G of the genes and a subset T of the experiments. Specifically, we are looking for *order-preserving submatrices* (OPSMs), in which the expression levels of all genes induce the same linear ordering of the experiments. It will be shown below that the OPSM search problem is NP-hard in the worst case. Such a pattern might arise, for example, if the experiments in T represent distinct stages in the progress of a disease or in a cellular process, and the expression levels of all genes in G vary across the stages in the same way.

9.1.3 OPSM Problem Definition

As we already know, the readout of a DNA chip containing n genes consists of n real numbers that represent the expression level of each gene, either as an absolute or as a relative quantity (with respect to some reference). When combining the readouts for m experiments (tissues),

[1]Based in part on a scribe by Igor Bogudlov and Vladimir Koushnir, Algorithms in Molecular Biology, Fall 2000.

each gene yields a vector of m real numbers. To make the OPSM results independent of the scaling of the data, we consider only the relative ordering of the expression levels for each gene, as opposed to the exact values. This motivates us to consider the permutation induced on the m numbers by sorting them. Thus we view the expressed data matrix, D, as an $n \times m$ matrix, where each row corresponds to a gene and each column to an experiment. The m entries in each row are a permutation of the numbers $\{1,\ldots,m\}$. The (i,j) entry is the rank of the readout of gene i in tissue j, out of the m readouts of this gene (see figure 9.1). Typical values for n and m are in the ranges $500 \leq n \leq 15000$ and $10 \leq m \leq 150$.

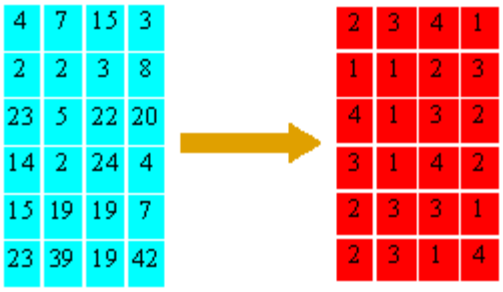

Figure 9.1: Ranks representation of a matrix.

We are seeking a biological progression that is represented as a "hidden" $k \times s$ submatrix $G \times T$ inside the data matrix D. The k genes from G are co-expressed in the s tissues from T. This means that the expression levels of all the genes in G move up and down together within the set T (see Figure 9.2)

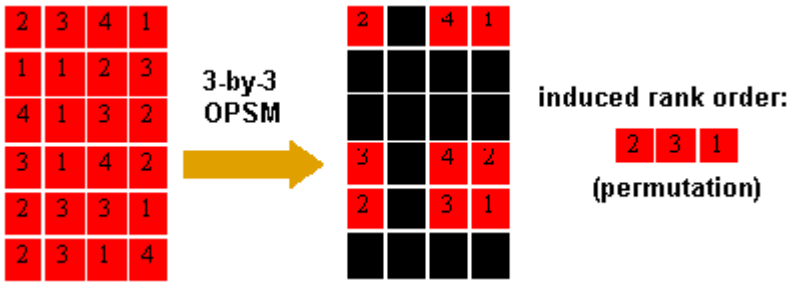

Figure 9.2: The left matrix is the ranks matrix from Figure 9.1. The red cells in the right matrix form a 3×3 OPSM. Please note that the 3 columns can be permutated such that in every row the ranks would be strictly increasing.

The computational task we address is the identification of large *order-preserving submatrices* (OPSMs) in an $n \times m$ matrix D. A submatrix is order-preserving if there is a permutation of its columns under which the sequence of values in every row is strictly increasing. In the case of expression data such a submatrix is determined by a set of genes G

and a set of tissues T such that within the set of tissues T, the expression levels of all the genes in G have the same linear ordering, as illustrated by Figure 9.3. Please note that these constraints are quite strict - for any two tissues, the same order relation is required over all the genes.

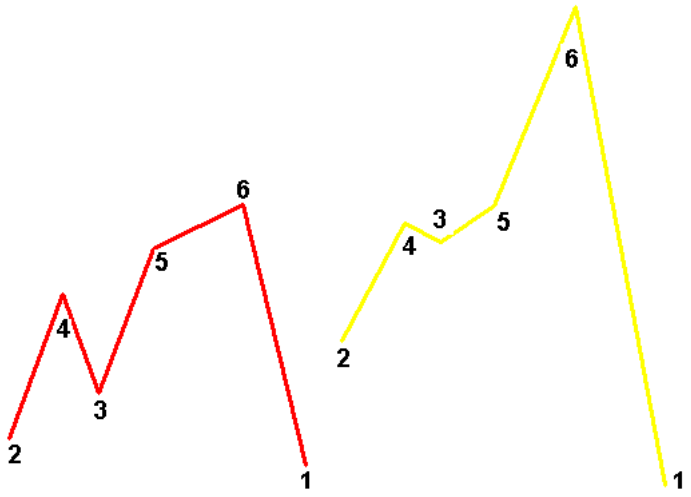

Figure 9.3: A graphical view illustrating the order-preserving constraints. The two graphs represent expression levels of 2 genes for 6 various tissues. It should be emphasized, that for every pair of tissues, the same order relation is required with respect to both genes.

9.1.4 OPSM Complexity

Problem 9.1 Let's formalize the OPSM problem, as presented in the previous section:
Input: An $n \times m$ ranks matrix M.
Question: Does M contain a $s \times t$ order preserving submatrix (OPSM) ?

Theorem 9.1 *The OPSM problem is NP-complete.*

Proof: A reduction from the Balanced Complete Bipartite Graph (BCBG) problem can be shown (the full reduction can be found in [4]). The BCBG problem is known to be NP-complete [9]. ■

It should be mentioned, however, that the above reduction assumes a "relatively square" matrix M, that is - k and s are close values. For the case $k \gg s$ the above reduction might not be sufficient.

9.1.5 Outline of the Solution

Having completed the presentation of the problem, we can now describe the algorithmic approach that was taken towards solving this problem. There are several stages we will go through:

- Generating simulated data, i.e. - planting an OPSM in an otherwise random matrix M. This is done using a specific probabilistic model, and is a means of evaluating the performance of the algorithm.
- Designing an OPSM hunting algorithm.
- Testing the algorithm on the simulated data.

9.1.6 Probabilistic Model

We model gene expression data by a random data matrix D in which an unknown order preserving submatrix $G \times T$ has been planted. The process of generating a data matrix with a planted order preserving submatrix (OPSM) consists of three stochastic steps:

1. First we choose at random the indices for the planted rows and columns.
2. Second, we order the planted columns randomly.
3. Finally, we assign ranks at random to the data matrix in a way which is consistent with the planted submatrix.

More formally, the parameters of the stochastic process are:

- n (number of genes).
- m (number of experiments).
- s (size of T).
- p (probability that a row i is in the subset G). One may ask why not simply choose k, as the size of G ? The answer is, that this would create an undesired dependence between the rows.

and the steps are:

1. Toss i.i.d. coins X_i for every i ($i = 1, \ldots, n$), with probability p of coming up heads (X_i=1). The set of genes G, is the set of indices i with X_i=1, and the expected size of G is pn. For the set of experiments, we choose a subset $T \subseteq \{1,\ldots,m\}$ of size s uniformly at random.

2. Pick uniformly at random a linear ordering t_1, t_2,..., t_s of the elements of T. Denote this linear ordering as π.

3. For every row i assign the m entries in the i-th row of D independently by a random permutation of 1,...,m.

4. For each row i with X_i=1 ($i \in G$), rearrange the ranks in the columns corresponding to T:
   The entry in column t_1, $D[i, t_1]$, will be assigned the lowest rank among these s entries, the entry in column t_2 will be assigned the second rank among the entries corresponding to T, and so on. The entry $D[i, t_s]$ will be assigned the highest rank among the entries of T.

With the completion of these steps the data matrix D with the planted submatrix $G \times T$ is determined. Note that in addition to the set of planted rows, G, every non-planted row has a probability of (1/s!) to satisfy the same ordering constraints as the planted rows. Given D and T, those "spurious planted" rows are indistinguishable from the "genuine planted" rows. Thus, the algorithmic goal is, for a given D, to recover the set of planted columns T, and their planted linear order π. The set of rows supporting this model ("genuine planted" together with the "spurious planted") is then uniquely defined.

9.1.7 OPSM Hunting Algorithm

Complete Models vs. Partial Models

Definition Let $T \subseteq \{1, \ldots, m\}$ be a set of size s. Let $\pi = (t_1, \ldots, t_s)$ be a linear ordering of T.The pair (T, π) is called a *complete OPSM model* or simply a *complete model*.

We say that a row $i \in \{1, \ldots, n\}$ supports (T, π) if the s corresponding entries, ordered according to the permutation π, are monotonically increasing, namely $D[i, t_1] < D[i, t_2] < \ldots < D[i, t_s]$. Intuitively, for a given t, we would like to find a complete model of size t, with a maximal number of supporting rows. Ideally, given t, one would like to find all the possible complete models. However, such an exhaustive search would be too computationally demanding.

Definition A *partial model* of order (a, b) specifies the indices of the a "smallest" elements $< t_1, \ldots, t_a >$ and the indices of the b "largest" elements $< t_{s-b+1}, \ldots, t_s >$ of a complete model (T, π). A partial model also specifies the size s of the complete model.

An alternative to the impractical exhaustive search mentioned above, would be to "grow" *partial models* iteratively, with the goal of "converging" to the best complete model. After every iteration, we will keep the l best partial models, l being a parameter chosen according

to performance limitations. It should be noted, that retaining only the best partial model instead of l models would decrease the chance for the algorithm to succeed.

Algorithmic Strategy

The outline of the algorithm is as follows:

- Initial stage - go over all partial models θ of order (1,1), each specifying two extreme columns and their order.
- Keep growing partial models iteratively.
- At each stage retain the l best partial models.
- Continue until complete models are reached.
- Output the best complete model found.

The complexity of this algorithm is $O(nm^2sl)$. It should still be explained how the l best partial models are chosen in every iteration, that is, how do we score a partial model.

Scoring a Partial Model

Given a *partial model* θ of order (a, b), we'd like to estimate its quality. Let us denote by τ the underlying (hidden) *complete model* used to generate the data matrix D, and let p denote the (unknown) probability for rows to be planted with respect to τ. To score the partial model θ, we assume that τ is an extension of θ, and estimate p. We use the estimated p as the quality measure of θ - the more rows seem to be planted with respect to θ, the more confident we are in θ. Thus, all we have to do is performing the actual estimation of p.

Let $D_\theta(i)$ denote the vector of ranks in the i-th row within the columns specified by the partial model θ.

Let

$$A_i = Prob[D_\theta(i)|X_i = 1], \tag{9.1}$$

and let

$$B_i = Prob[D_\theta(i)|X_i = 0]. \tag{9.2}$$

That is, A_i denotes the probability of observing the ranks $D_\theta(i)$, given that i is a planted row, and B_i denotes the probability of observing $D_\theta(i)$, given that row i is not planted.

Note, that A_i is an increasing function of $(D[i, \pi(t-b+1)] - D[i, \pi(a)])$, i.e. , the gap between the smallest among the "largest" elements and the largest among the "smallest" elements. Intuitively, this gap indicates how many extension options exist for θ. Interestingly, fact is, since the computation depends on only two values, it doesn't become complicated when the number of columns included in the current partial model increases. As was explained above, we are interested in the probability that a given row is planted (by a complete model extending θ) after observing $D_\theta(i)$. This probability can be computed using Bayes Theorem:

$$Prob[X_i = 1 | D_\theta(i)]) = \frac{A_i p}{A_i p + B_i(1-p)} \tag{9.3}$$

remove We denote this quantity by $p_i(\theta, p)$. Summing over all rows, we get that the expected number of planted rows (by a complete model that extends θ) after observing D_θ, is $\sum_{i=1}^{n} p_i(\theta, p)$. Let $X = \sum_{i=1}^{n} X_i$ be the random variable counting the number of planted rows, and let Y be the random variable consisting of all the data for the partial model θ. Then $E[X] = np$ and, as we have just seen, $E[X|Y] = \sum_{i=1}^{n} p_i(\theta, p)$. Using the identity $E[X] = E[E[X|Y]]$ we find that $E[\sum_{i=1}^{n} p_i(\theta, p)] = np$. Thus $\sum_{i=1}^{n} p_i(\theta, p)$ is an unbiased estimator of np. Moreover, application of a Chernoff bound shows that the random variable $\sum_{i=1}^{n} p_i(\theta, p)$ is concentrated around its expectation, and therefore is a reliable estimator of np. Thus we can estimate p by solving for p the implicit equation $\sum_{i=1}^{n} p_i(\theta, p) = np$. Recalling the formula for $p_i(\theta, p)$ we get the implicit equation

$$\sum_{i=1}^{n} \frac{A_i p}{A_i p + B_i(1-p)} = np.$$

Thus, the expected number of planted rows, given the data in θ columns is

$$\sum_{i=1}^{n} \frac{A_i p}{A_i p + B_i(1-p)} \tag{9.4}$$

It can be shown that p can be estimated by solving the following equation:

$$np = \sum_{i=1}^{n} \frac{A_i p}{A_i p + B_i(1-p)} \tag{9.5}$$

This equation can be solved numerically, giving a single positive solution for p. For full details about the estimation of p see [4]. This concludes the computation of p, and given a partial model θ we are able to compute its score.

9.1.8 Algorithm Results on Simulated Data

The algorithm was tested on simulated data. All the simulated datasets were 1000×50 matrices. The number of planted columns s varied over five values $s = 3, 4, 5, 7, 10$. The

number of rows in G was determined by flipping a p biased coin per row, where p varied over four values $p = 0.025, 0.05, 0.075, 0.1$. Table 9.1 reports the probabilities of the algorithm correctly recovering the set of planted columns, and their correct internal order. Each entry of the table is based on one hundred random datasets.

$p \quad \backslash \quad s$	3	4	5	7	10
0.025	0.0	0.01	0.21	0.72	0.92
0.05	0.17	0.7	0.94	1.0	1.0
0.075	0.69	0.98	1.0	1.0	1.0
0.1	0.92	1.0	1.0	1.0	1.0

Table 9.1: Probability of identifying the planted columns (in the correct order). For all experiments $n = 1000, m = 50$ and $\ell = 100$. Probabilities are based on 100 simulations per entry.

In cases where the algorithm fails to recover the planted submatrix, the statistical significance of the recovered submatrix was compared to the significance of the planted submatrix. For certain values of the parameters, the algorithm recovered a submatrix larger than the planted one. Clearly, those cases should not be considered as failures of the algorithm, but rather serve as indication that for those simulation parameters there is no hope to recover the planted submatrix.

9.1.9 Algorithm Results on Real Data

In addition to simulation tests, the algorithm was also run on one breast tumor dataset [5]. This dataset contains 3226 genes and 22 tissues. Out of these 22 tissues there are 8 with brca1 mutations, 8 with brca2 mutations, and 6 sporadic breast tumors.

Several statistically significant OPSMs were found in the dataset. One such OPSM had $s = 4$ tissues, supported by $k = 347$ genes. This pattern is statistically significant since we would expect to find only $(3226/4!) = 134$ genes that support such a pattern at random, and the overabundance of supporting genes might suggest biological relevance. Interestingly, the first three tissues are all brca2 mutations, while the last one (largest expression levels) is sporadic. These results are graphically depicted in Figure 9.4. The region marked by thick lines indicates the 347×4 order preserving submatrix.

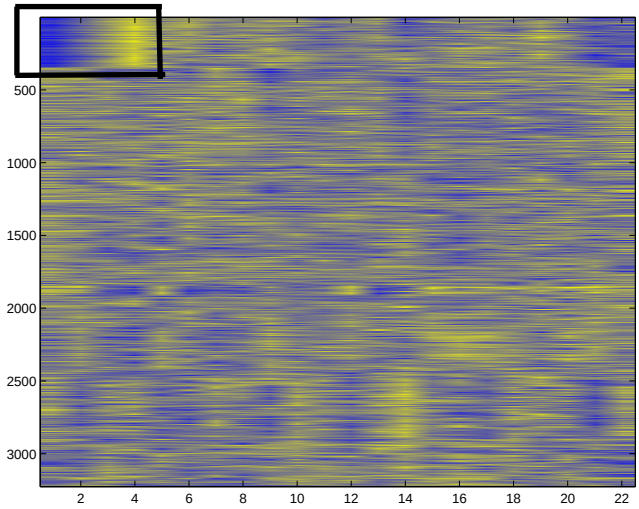

Figure 9.4: An order preserving submatrix identified in breast cancer data, consisting of 347 genes and 4 tissues.

9.2 Genetic Networks

9.2.1 Preface

An ultimate goal of a molecular biologist is to use genetic data to reveal fundamental cellular processes, and their impact on complex organisms. In order to achieve this goal one has to study complex systems of several genes and proteins, that carry out a specific function, rather than single genes. Such research can help identify proteins and genes associated with human diseases, which has huge practical importance.

9.2.2 Biological Background [3]

The different cell types in a multicellular organism differ dramatically in both structure and function. They become different from one another because they synthesize and accumulate different sets of RNA and protein molecules.

Studies of the number of different mRNA sequences in a cell suggest that a typical higher eucaryotic cell synthesizes 10,000 to 20,000 different proteins. Most of these are too rare to be detected by two-dimensional gel electrophoresis of cell extracts. If these minor cell proteins differ among cells to the same extent as do the more abundant proteins, as is commonly assumed, a small number of protein differences (perhaps several hundred) is sufficient for creating vast differences in cell morphology and behavior.

Gene Expression Regulation

If differences between the various cell types of an organism depend on the particular genes that the cells express, at what level is the control of gene expression exercised ? There are many stages in the pathway leading from DNA to protein, and all of them can, in principle, be regulated. Thus a cell can control the proteins it synthesizes in several ways (see also Figure 9.5):

1. Controlling when and how often a given gene is transcribed (transcriptional control).
2. Controlling how the primary RNA transcript is spliced or otherwise processed (RNA processing control).
3. Selecting which completed mRNAs in the cell nucleus are exported to the cytoplasm (RNA transport control).
4. Selecting which mRNAs in the cytoplasm are translated by ribosomes (translational control).
5. Selectively destabilizing certain mRNA molecules in the cytoplasm (mRNA degradation control).
6. Selectively activating, inactivating, or compartmentalizing specific protein molecules after they have been made (protein activity control).

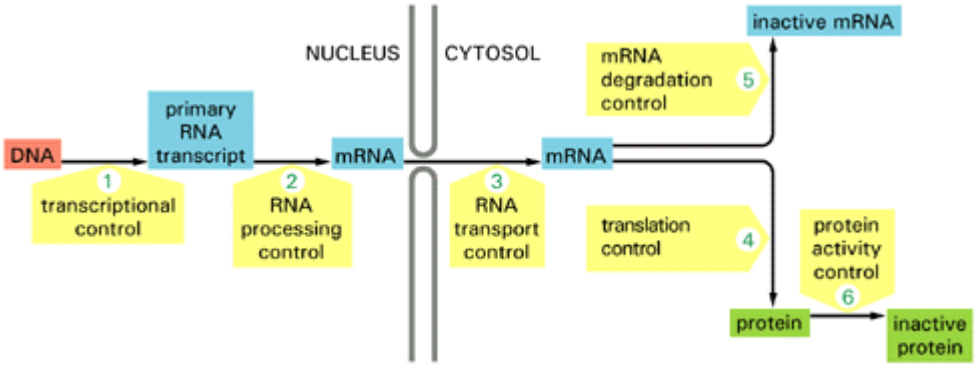

Figure 9.5: Six stages at which eucaryote gene expression can be controlled.

For most genes transcriptional controls are paramount. This makes sense because, of all the possible control points illustrated in Figure 9.5, only transcriptional control ensures that no superfluous intermediates are synthesized.

DNA-binding Motifs in Gene Regulatory Proteins

How does a cell determine which of its thousands of genes to transcribe? The transcription of each gene is controlled by a regulatory region of DNA near the site where transcription begins. Some regulatory regions are simple and act as switches that are thrown by a single signal. Other regulatory regions are complex and act as tiny microprocessors, responding to a variety of signals that they interpret and integrate to switch the neighboring gene on or off. Whether complex or simple, these switching devices consist of two fundamental types of components:

- Short stretches of DNA.
- Gene regulatory proteins that recognize these sequences and bind to them.

These proteins are called *transcription factors*. A transcription factor can be an *activator* (activating and accelerating the transcription) or a *repressor* (repressing the transcription). Additionally, a group of genes can be regulated simultaneously (the precise mechanisms differ between eucaryotes and procaryotes). There are several additional mechanisms for controlling gene expression levels, e.g., the extent to which the DNA is folded (in eucaryotes).

9.2.3 Genetic Networks

Definition A *genetic network* is a set of molecular components such as genes, proteins and other molecules, and interactions between them that collectively carry out some cellular function.

Genetic Networks describe functional pathways in a given cell or tissue, representing processes such as metabolism, gene regulation, transport and signal transduction. Let us examine several examples:

1. **Expression of the Gene proB**

   Figure 9.6 depicts the gene's expression and its role in catalyzing a specific chemical reaction in the cell. The proB gene is being expressed into the gamma-glutamyl-kinase protein, which catalyzes a reaction involving glutamate and ATP, that produces gamma-glutamyl-phosphate and ADP compounds.

2. **A Simple Metabolic Pathway - Proline Biosynthesis**

   The next example is part of a simple metabolic pathway, involving a chain of generated proteins, which is shown on Figure 9.7. One of the final products of the chain,

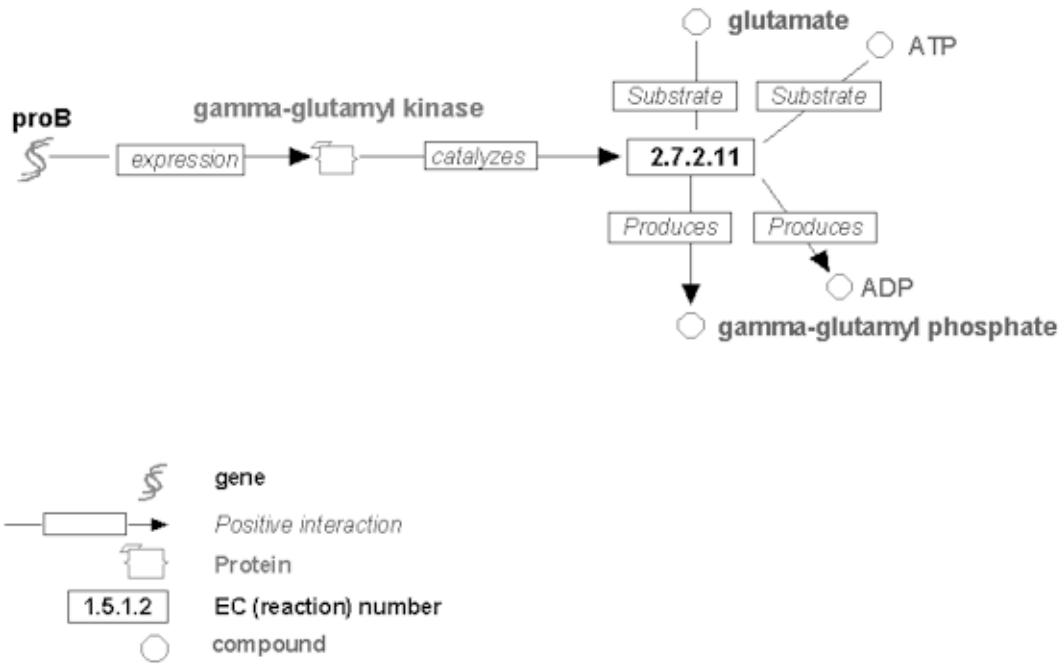

Figure 9.6: An example of the role of gene expression in catalyzing chemical reactions.

proline, inhibits the initial reaction that started the whole process. This "feedback inhibition" pattern is highly typical to genetic networks, and serves to regulate the process execution rate.

3. **Methionine Biosynthesis in E-coli.**

The following two figures show a more complex genetic network, describing Methionine biosynthesis in E-coli. The second figure is a schematic representation of the pathway, with most nodes omitted, but it can give a better idea of the overall topology.

4. **A Genetic Network that Performs Signal Transduction**

This last example, depicted in Figure 9.10, is that of signal transduction - a complex cellular process initiated by a signaling protein, arriving from outside of a cell. This process eventually affects gene expression in both the cytoplasm and inside the nucleus.

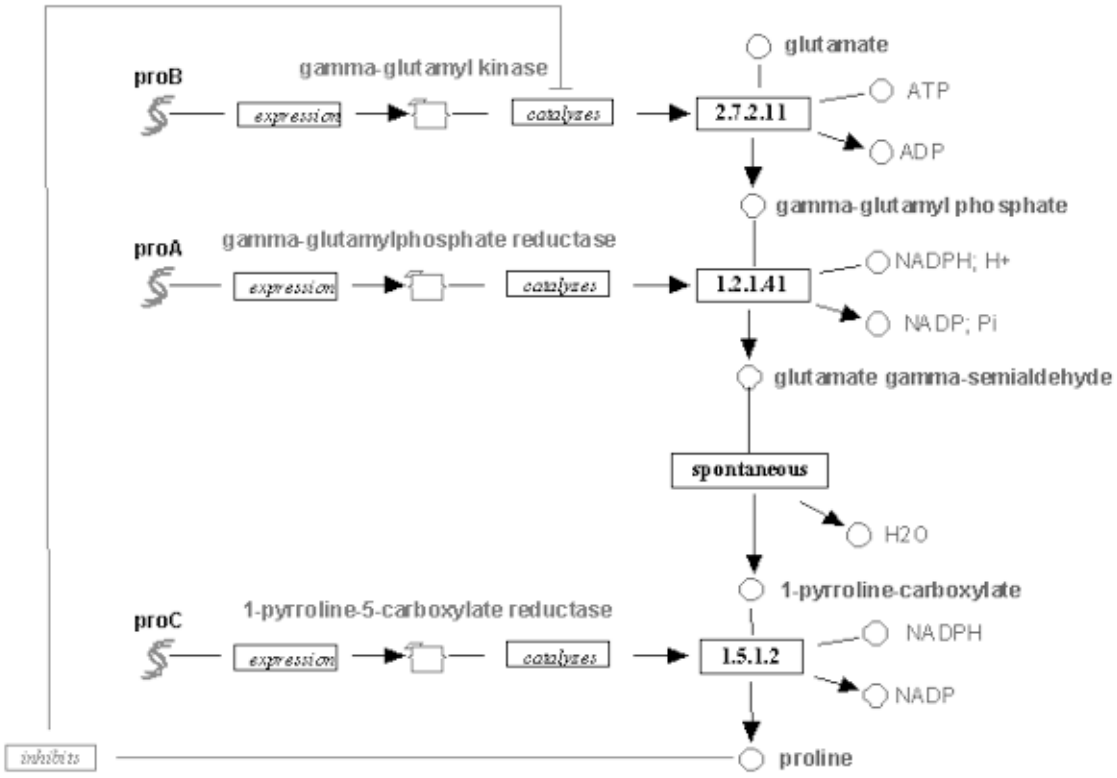

Figure 9.7: An example of a metabolic pathway: Proline biosynthesis.

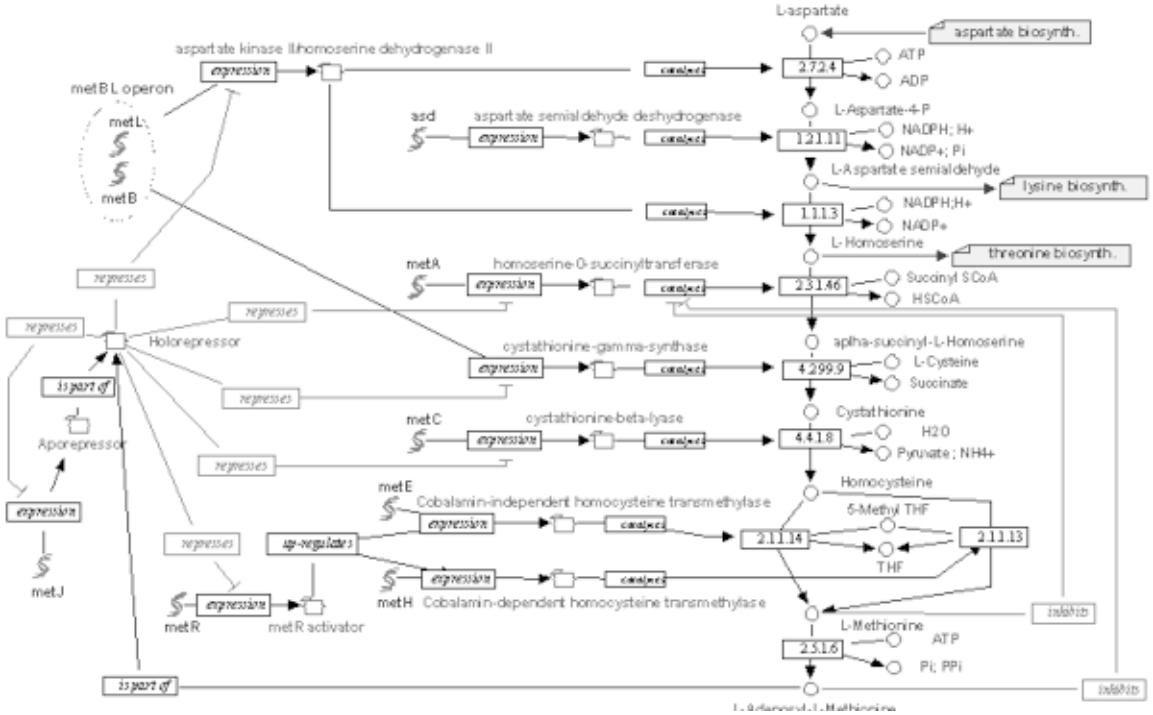

Figure 9.8: Methionine biosynthesis network in E-coli.

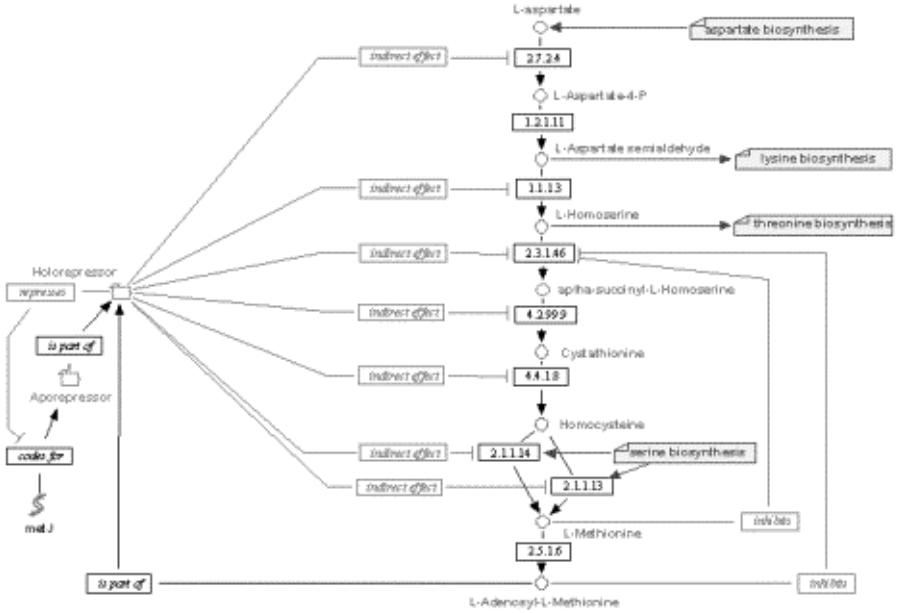

Figure 9.9: Schematic representation of the biosynthesis pathway presented in Figure 9.8.

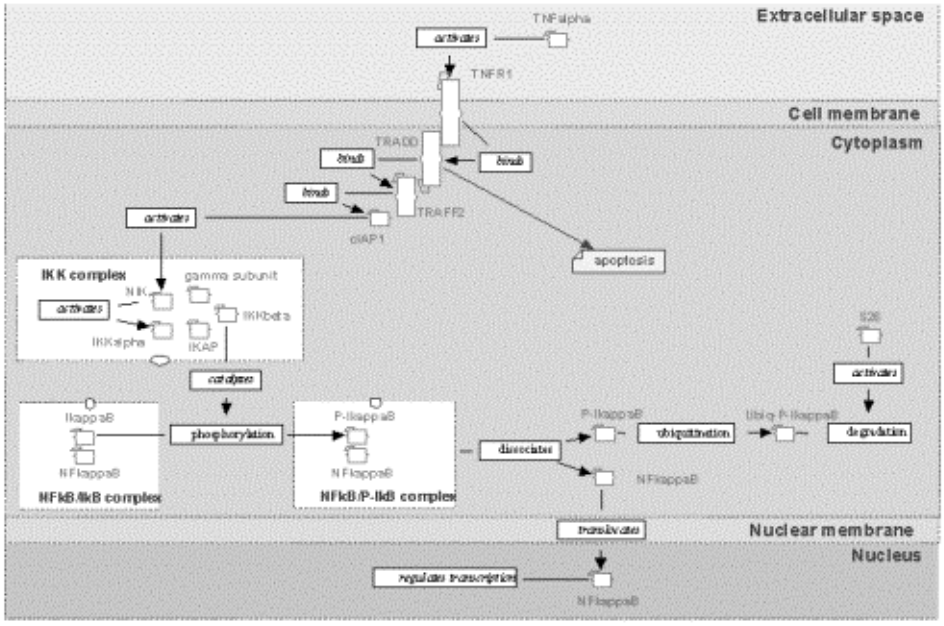

Figure 9.10: A genetic network that performs signal transduction from outside the cell into the nucleus.

9.2.4 Functional Analysis

Using a known structure of such networks it is sometimes possible to describe the behavior of cellular processes, reveal their function and determine the role of specific genes and proteins in them. That is why one of the most important and challenging problems today in molecular biology is that of *functional analysis* - discovering and modelling Genetic Network from experimental data.

Biological Tools

Addressing this problem has been made possible by recent advances in genetic sequencing and development of a whole new generation of sophisticated biological tools.

The most promising technique to date is based on the view of gene systems as a logical network of nodes that influence each other's expression levels. Consequently, one may obtain some information about gene interactions in the network by measuring gene expression levels.

A variety of experimental tools have been developed recently with the ability to observe the expression of many genes simultaneously. At the forefront of these technologies lies the DNA microarray, commonly used to monitor gene expression at the level of mRNA. Similarly, the rapid identification of proteins and their abundances is becoming possible through methods such as 2D polyacrylamide gel electrophoresis [1], 2-hybrid systems [2],

protein chips [14], etc. The main contribution of all of these technologies is that numerous genes can be monitored simultaneously, making it possible to perform a global expression analysis of the entire cell.

Additional information about a genetic network may be gleaned experimentally by applying a directed perturbation to the network, and observing expression levels of every gene in the network, in the presence of the perturbation. Perturbations may be *genetic*, in which the expression levels of one or more genes are fixed by *knockout* (removal of the gene) or *overexpression* (higher than usual level of gene expression), or *biological*, in which one or more non-genetic factors are altered, such as a change in environment, nutrition, or temperature. Such biological experiments are very costly and very few such perturbations may be performed at one time. Thus, reducing the number and cost of experiments is crucial.

The methods presented above supply biological data in terms of expression levels of many genes at different time points and under various conditions. The functional analysis of the data can be defined as a computational problem, aiming to infer some plausible model of the network from the observations, while keeping the number or cost of biological experiments at a minimum. The model should describe how the expression level of each gene in the network depends on external stimuli and expression levels of other genes. Additional goals include construction of a knowledge-base of gene regulatory networks, and verification of pathways or genetic network hypotheses.

9.2.5 Genetic Network Models

Several models have been proposed in the literature to capture the notion of genetic networks and allow mathematical solutions of the computational problem of modelling biological processes:

- **Linear Model:**
  This model, proposed by D'haeseleer et al [7, 6], assumes that the expression level of a node in a network depends on a linear combination of the expression levels of its neighbors.

- **Boolean Model:**
  Proposed by Liang, Fuhrman and Somogoyi [12]. It assumes only two distinct levels of expression - 0 and 1. According to this model, the value of a node at time $t + 1$ is a boolean function of the values of its neighbors at time t.

- **Bayesian Model:**
  Proposed by Friedman et. al [8]. It attempts to give a more accurate model of network behavior, based on Bayesian probabilities for expression levels.

Therefore, we concentrate on the Boolean model. The Bayesian Model will be discussed in detail in the next Lecture.

9.2.6 Boolean Network Model

According to the boolean model, a network is represented by a directed graph $G = (V, F)$, where:

- V represents nodes (elements) of the network.
- F ia a set of boolean functions (see below), that defines a topology of edges between the nodes.

A node may represent either a gene or a biological stimulus, where a stimulus is any relevant physical or chemical factor which influences the network and is itself not a gene or a gene product. Each node is associated with a steady-state expression level x_v, representing the amount of gene product (in the case of a gene) or the amount of stimulus present in the cell. This level is approximated as high or low and is represented by the binary value 1 or 0, respectively.

Network behavior over time is modelled as a sequence of discrete synchronous steps. The set $F = \{f_v | v \in V\}$ of boolean functions assigned to the nodes defines the value of a node in the next step, depending on values of other nodes, which influence it. The functions f_v are uniquely defined using truth tables. An edge directed from one node to another represents the influence of the first gene or stimulus on that of the second. Thus, the expression level of a node v is a boolean function f_v of the levels of the nodes in the network which connect (have a directed edge) to v.

Definition A *trajectory* is a sequence of consecutive states of the network. It can be viewed as a list of N-dimensional vectors (N being the number of nodes in the network), each representing a state.

9.2.7 A Complementary Approach

We can view an organism as a very large genetic network. If we knew all the interactions of such a network, we could perfectly understand every single detail in the organism. That is, we could understand which genes, proteins and other molecules are involved in every biological process, how exactly the process takes place, etc.

This might be the ultimate goal of biological science, but obviously we are light years away from it.We therefore make several simplifying assumptions.We model the organism as many distinct genetic networks, which loosely interact among themselves. We further assume that every gene depends on no more than 5 other genes.

Indeed, these are heavy assumptions, but they are necessary in order for genetic networks to be useful in modelling biological processes.

Given such a group of genetic networks, we can explore their properties (global structural features, types of possible dynamic behaviors etc.).

Definition An *ensemble of genetic networks* is composed of similar networks that share some features. The non constrained features vary at random between networks in the ensemble.

Properties of an Ensemble of Networks:

- Every network consists of N nodes (genes).
- Each gene is influenced directly by exactly k other input genes.
- For each node, the k input genes are chosen at random.
- For each node, its boolean function is chosen at random from the 2^{2k} possible functions.

9.2.8 Simplified Description

Following are a few assumptions taken in order to simplify the model:

- The activation of genes depends on proteins and chemicals.
- The synthesis of proteins participating in a regulatory process is very fast compared to the regulatory process itself.
- Regulatory proteins decay much faster than the duration of the regulatory processes.
- The concentrations of the regulatory chemicals are constant.
- We can express the activation level (mRNA level or protein level) in time $t + \delta t$ as a function of the activation at time t. We will later use $\delta t = 1$.
- Loss of memory occurs within δt time, that is, knowledge of steps before time T is not needed.

9.2.9 Kauffman's Model

Kauffman's Model [10, 11]uses boolean gene levels, 1 for active and 0 for inactive. It also assumes that time $t + 1$ is determined by a boolean function of the levels of a fixed set of input genes at time t. This means it can use only 1-step memory. All updates are executed in a deterministic way and are synchronized. External chemicals are not explicitly taken into account.

Kauffman's Model is dynamic:

- At time 0, a level is given to every gene.

- At each time step $t = 1, 2...$ every gene has a level $x_i(t)$, which is determined according to the boolean functions.

- The global state of the system is $X = [x_1, x_2, ...x_n]$ and we say that $X(t)$ alone determines $X(t+1)$. As time passes, the system moves from state $X(t)$ to $X(t+1)$, $X(t+2)$ and so on, following a trajectory.

The states can be thought of as corners in the unit hypercube and a step from one global state to another can be thought of as shifting from one corner to another. Note, that a legal move does not have to be between two adjacent corners, since adjacent corners differ only by one bit.

Examples

Figure 9.11 gives an example of a simple boolean network and associated truth tables. This example shows a network of three nodes - a, b and c. As one can see, the expression of c directly depends on the expression of b, which in turn directly depends on a. Note that b influences more than one node, a and c (**"pleiotropic regulation"**), and that a is influenced by more than one node (**"multigenic regulated"**).

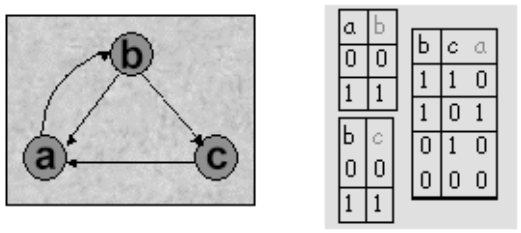

Figure 9.11: source: [13].A sample boolean network.

The assignment of values to nodes fully describes the *state* of the model at any given time. The change of model state over time is fully defined by the functions in F. Initial assignment of values uniquely defines the model state at the next step and consequently, on all the future steps. Thus, the network evolution is represented by its *trajectory*.

Figure 9.12 shows two such trajectories for the sample network. Since the number of possible states is finite, all trajectories eventually end up in single *steady state*, or a cycle of steady states.

Definition An *attractor* of a trajectory is a single steady state, or a cycle at the end of the trajectory. *The basin of attraction* for a specific attractor is the set of all trajectories leading to it.

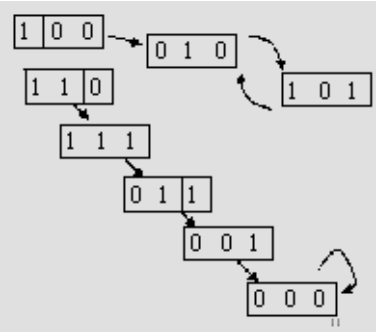

Figure 9.12: source: [13].States trajectories.

One or more attractors are possible. The network in our example has two attractors - one is the steady state $(0,0,0)$, and the other is a cycle $(0,1,0) \leftrightarrow (1,0,1)$. The attractors are reached when $t \to \infty$. In a finite boolean network, one of the attractors is reached in a finite time.

States in genetic networks are often characterized by *stability* - "slight" changes in value of a few nodes do not change the attractor. Biological systems are often *redundant* to ensure that the system stays stable and retains its function even in the presence of local anomalies. For example, there may be two proteins, or even two different networks with the same function, to backup each other.

9.2.10 Ensembles of Networks

We defined above what an *"ensemble of networks"* is, and which properties it possesses. However, each network has its own dynamics. The main features of the model, attractors and basins, are determined by the degree of connectivity in each network. A degree of connectivity k means that the in-degree of each node is exactly k.

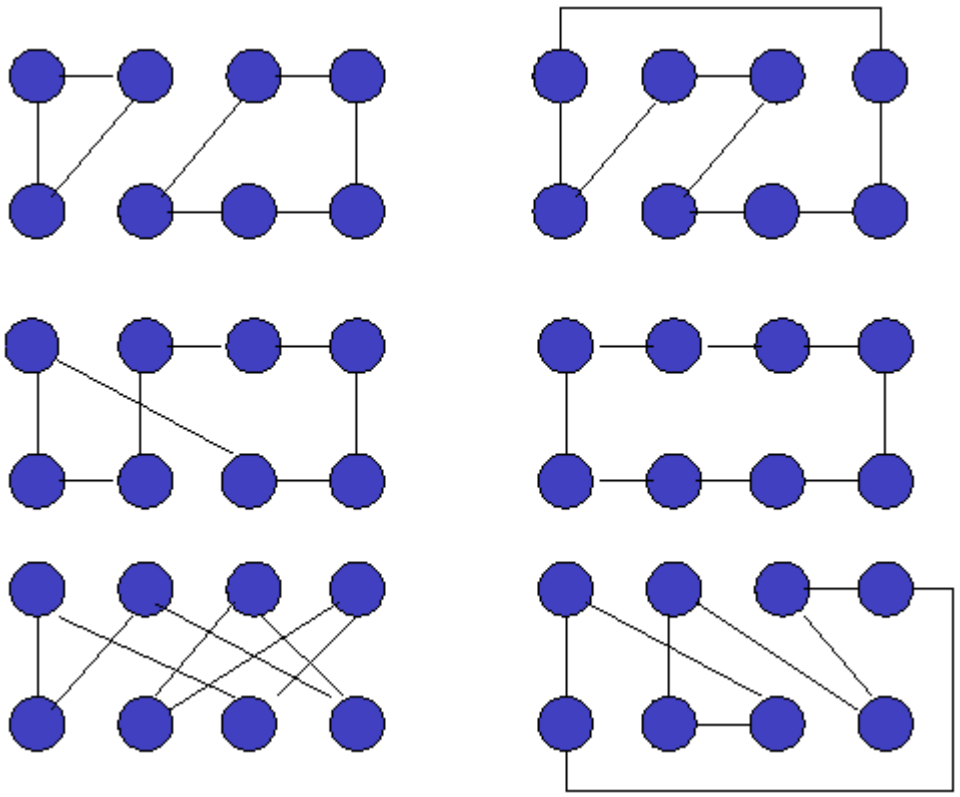

Figure 9.13: An ensemble of random networks with ($k = 2$). Note that every node in every network has degree 2.

High In-Degree

In the case that k is as high as $N-1$:

- $X(t+1)$ is completely uncorrelated to $X(t)$, the output associated to each input set is random. There is no correlation between outputs corresponding to two inputs which differ even by a single bit. The system is chaotic and the homeostatic stability is very low, nearby initial states go to different attractors, and changing one input function completely destroys the basin structure.
- The number of attractors, about N/e, is very small compared to the 2^N possible states.
- The cycles are huge, period size is around $2^{0.5N}$.

For example, for $N = 100,000$ we get $10^{30,000}$ states, only 37,000 attractors and cycles are as long as $10^{15,000}$.

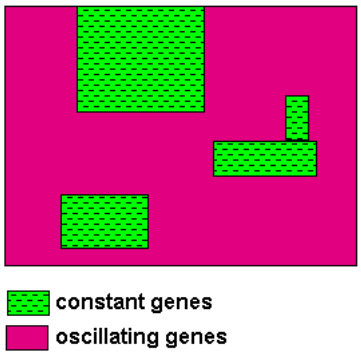

Figure 9.14: A 2-dimensional lattice view of a generic network, i.e., every cell in the lattice represents a gene. It can be seen, that when the in-degree is high, most of the genes are oscillating, that is, their state changes very frequently, and only few genes reach a constant state. Furthermore, the oscillating genes form a giant component, instead of being scattered all over the lattice.

Low In-Degree

In the case of $k = 2$:

- Basins are regular: nearby initial states usually reach the same attractor, high homeostatic stability, spontaneous order, even though inputs and functions are completely random.
- The number of attractors is relatively high - about $N^{1/2}$.
- Average cycle length is $N^{1/2}$.

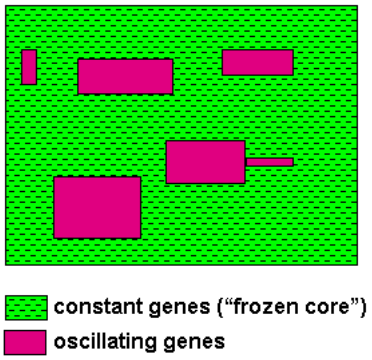

Figure 9.15: A 2-dimensional lattice view of a generic network with low in-degree. One can see that the effect is opposite to that observed in Figure 9.14 - most of the genes are constant, forming a giant component, while only few genes oscillate.

9.2.11 Concluding Remarks about Kauffman's Model

Kauffman's model is a highly idealized representation of real genetic networks, due to the following reasons:

- Chemicals are not taken into account.
- Regulatory proteins are assumed to be synthesized very fast with respect to the regulation process itself.
- Synchronous activation may introduce "spurious cycles" in boolean dynamical systems.
- Fixed in-degree k is assumed for all genes.

However, Kauffman's model allows us to address issues which would otherwise be neglected, and to develop an appropriate language in which we can formulate key questions, such as:

- The importance of attractors in determining the properties of genetic networks.
- Robustness and basins of attraction.
- The importance of the average degree of connectivity.

Kauffman's model also allows us to examine in a new way the interplay between selection and self-organization. Moreover, it demonstrates the importance of studying ensembles of networks to gain insight about their generic properties.

Bibliography

[1] http://www.expasy.ch/ch2d/.

[2] http://www.uib.no/aasland/two-hybrid.html.

[3] B. Alberts, D. Bray, J. Lewis, M. Raff, K.Roberts, and J.D. Watson. *Molecular Biology of the Cell*. Garlend Publishing Inc., New York and London, 1994.

[4] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. In *The Sixth Annual International Conference on Research in Computational Molecular Biology RECOMB 2002*, 2002.

[5] Y. Chen, M. Radmacher, M. Bittner, R. Simon, P. Meltzer, B. Gusterson, M. Esteller, M. Raffeld, Z. Yakhini, A. Ben-Dor, E. Dougherty, J. Kononen, L. Bubendorf, W. Fehrle, S. Pittaluga, S. Gruvberger, N. Loman, O. Johannsson, H. Olsson, B. Wilfond, G. Sauter, O.P. Kallioniemi, A. Borg, J. Trent, I. Hedenfalk, and D. Duggan. Gene-expression profiles in hereditary breast cancer. *NEJM*, 344:539–548, 2001.

[6] P. D'haeseleer and S. Fuhrman. Gene network inference using a linear, additive regulation model. *Bioinformatics*, 2000.

[7] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. In *Pacific Symposium on Biocomputing*, pages 41–52, Hawaii, Hawaii, 1999. World Scientific Publishing Co.

[8] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *J. Computational Biology*, 7(3):601–620, Nov 1998.

[9] 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.

[10] S.A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[11] S.A. Kauffman. *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford University Press, 1995.

[12] S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Symposium on Biocomputing*, pages 18–29, Maui, Hawaii, 1998. World Scientific Publishing Co.

[13] R. Somogyi and C. Sniegoski. *Complexity*, 1:45–63, 1996.

[14] H. Zhu, J.F. Klemic, S. Chang, P. Bertone, A. Casamayor, K.G. Klemic, D. Smith, M. Gerstein, M.A. Reed, and M. Snyder. Analysis of yeast protein kinases using protein chips. *Nature Genetics*, 26:283–289, 2000.