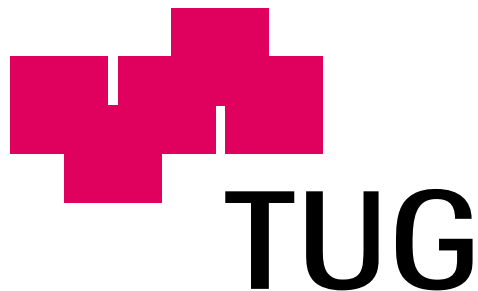


Dejori Mathäus

Analyzing Gene-Expression Data with Bayesian Networks

Diplomarbeit



Institut für Elektro- und Biomedizinische Technik
Technische Universität Graz
Inffeldgasse 18, A-8010 Graz
Vorstand: Univ.-Prof. Dipl.-Ing. Dr.techn. Gert Pfurtscheller

Betreuer: Priv. Doz. Dipl. Phys. Dr. rer. nat. Martin Stetter, Siemens AG,
München
Begutachter: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Trajanoski Zlatko

Graz, Juni 2002

**for my family
especially for my brother Michael:
"welcome back"**

Kurzfassung

Genetische Netze können helfen Wechselwirkungen zwischen Genen zu erforschen und damit die Funktion einzelner Gene zu identifizieren.

Ziel dieser Diplomarbeit war es ein Verfahren zum Strukturlernen Kausaler Netze zu formulieren und speziell für die Anwendung an Microarraydaten zu entwickeln. Zunächst wurde das Verfahren an Benchmark-Daten mit bekannter statistischer Struktur angewandt. Danach folgte die Applikation des Strukturlernverfahrens an realen Genexpressionsdaten von Spellmann et al. mit dem Ziel statistische Abhängigkeiten oder kausale Zusammenhänge zwischen verschiedenen Genen zu schätzen. Die gelernten Netze können als statistische Schätzer für die Struktur genetischer Netze verstanden werden, wobei die Knoten einzelnen Genen entsprechen und die Struktur kausale Abhängigkeiten modelliert. Dadurch kann auf die Funktion einzelner Gene, wie z.B. in Bezug auf die Regulation der Transcription anderer Gene oder in Bezug auf metabolische Prozesse, geschlossen werden.

Schliesslich wurden die Ergebnisse mit denen anderer statistischer Methoden und mit dem Wissen aus der Molekularbiologie verglichen. Die gelernten Netze stimmen einerseits mit den Ergebnissen anderer statistischer Methoden überein und zeigen auf der anderen Seite neue, noch unbekannte Funktionen und Abhängigkeiten zwischen Genen auf.

Schlüsselwörter: Kausale Netze, Gen-Netzwerke, Microarrays, Maschinelles Lernen, Functional Genomics.

Abstract

Genetic networks can help identifying interactions between genes as well as the role of individual genes for the cellular function.

In this thesis a framework for learning Bayesian networks was developed to learn network structures out of microarray datasets. Before it was applied to such biological data various tests are made to benchmark the implementation. Additionally the Bayesian framework was tested for its convergence and reliability properties for sparse high-dimensional datasets such as microarray measurements.

Then structures were learned from the well known cell-cycle microarray dataset of Spellman et al. Since the learned structures can be interpreted as genetic networks, where nodes correspond to genes and the graph structure encodes causal relationships among them, such structures can be used to analyze the function of several genes e.g. their role in transcription regulation or in metabolic processes. To get a feedback, the learned structures were finally compared with results from other statistical approaches and with knowledge from biology. Some features of the learned networks are similar to the results from other statistical methods, whereas other features reveal new functions and relationships among genes, which are often biologically reasonable.

Keywords: Bayesian networks, genetic networks, microarray, machine learning, functional genomics.

Contents

1	Introduction	7
1.1	Objective	8
2	Methods	9
2.1	Bayesian network	9
2.1.1	D-separation	10
2.1.2	Structure equivalence	11
2.2	Structural learning	12
2.2.1	Scoring function	12
2.3	Heuristic methods	15
2.3.1	Local search	15
2.3.2	Greedy search	15
2.3.3	Simulated annealing	16
2.4	Missing values	17
2.4.1	Expectation maximization	18
2.4.2	Structural expectation maximization	18
2.5	Gene expression	19
2.6	Transcriptional regulation	20
2.6.1	Structure of a gene	20
2.6.2	The transcription initiation complex	21
2.6.3	Regulatory proteins	21
2.6.4	Tumor suppressor genes	22
2.7	Cell cycle regulation	23
2.8	DNA microarrays	24
2.9	Modelling gene regulatory mechanisms	25
2.10	Implementation	26
3	Results	27
3.1	Benchmark experiments	27
3.1.1	Greedy search vs. simulated annealing	27
3.1.2	Various sample sizes	34

<i>CONTENTS</i>	5
3.2 Microarray experiments	35
3.2.1 The microarray experiment by Spellman et al.	35
3.2.2 The microarray-dataset	37
3.2.3 Learned subnetworks	38
4 Discussion	46
4.1 Future work	47
4.2 Conclusion	48
4.3 Acknowledgments	48
5 Benchmark-Datasets	49
5.1 The Asia-network	49
5.2 The Alarm-network	51

Chapter 1

Introduction

Until recently, the attention of biologists have focused largely on the elucidation of the function and structure of genes and of gene families.

Today, after the Human Genome project (HGP) [24, 40] has finally achieved and reported about 90% of human genome sequence, molecular biology research emphasizes understanding the mechanisms of interactions between ensembles of genes and proteins within *regulatory networks*. Thanks to high throughput technologies, with their capacity for collecting data on a genome-wide scale within a short time period, a huge amount of biological data is available. For example DNA microarrays provide a global, high throughput approach to discovering which genes are on and off in a cancer cell, in other words which genes or gene clusters are involved in cancer initiation. But to extract the relevant information from such large-scale data, intensive computational interpretations and evaluations are needed.

The goal of *computational genomics* is to filter out the relevant information hidden in the datasets generated by DNA sequencing, DNA microarrays, and other kinds of high-throughput genomics technology. Since the data of such high-throughput experiments are highdimensional, noisy and sparse in sample size especially statistical methods play an important role in the their interpretation by finding trends and patterns in the experimental results [52, 2].

Previous efforts at modeling genetic regulatory networks from highdimensional datasets have generally fallen into one of three classes, either employing Boolean networks [47, 13, ?], which are restricted to logical relationships between variables, or using systems of differential equations [6, 42] to model the continuous dynamics of coupled biological reactions. The work of Friedman et al. [16] uses Bayesian networks to analyze expression data.

The statistical framework of *Bayesian learning* is designed for domains with a large number of variables and for handling noisy data, since it deals with uncertainty [38]. Another advantage of this probabilistic approach is the ability to

combine prior knowledge with the information extracted from data. A *Bayesian network* in particular is a probabilistic model which describes the multivariate probability distribution for a set of variables. The basic idea is to display the associations among the variables, namely the (*conditional dependencies* and *independencies*), by means of a graph. Thus, Bayesian networks belong to the class of *graphical models* [35]. *Learning Bayesian networks* [21, 49] use Bayesian statistics to find the network structure and the corresponding model parameters which describe best the probability distribution for which the dataset is drawn. The goal of learning Bayesian networks from highdimensional datasets, generated from microarray experiments, is to reveal relationships between various genes, by extracting the information about their dependencies and independencies encoded in the dataset and visualizing this relationships by a network structure, which is interpreted as a *genetic network*. Such a resulting network may be a basis to reveal new functions of genes and their proteins in combination with other experiments.

1.1 Objective

In this thesis a framework should be developed to learn Bayesian networks from highdimensional and noisy datasets, and to visualize the learned network structure in such a manner that the results are easy to understand.

Before analyzing real microarray datasets, the framework should be tested on artificial data to examine different learning algorithms, and on artificial datasets, which have similar properties as microarray datasets, namely large in variable size but small in sample size. This should be done to get a feeling of how informative is the statistical content which can be obtained from datasets of high throughput experiments.

Then structures should be learned from a microarray dataset applied to the yeast *S. cerevisiae* during cell cycle from Spellmann et al. [45], and finally, they should be compared with results from other statistical approaches [45, 16] and with knowledge from molecular biology.

Chapter 2

Methods

2.1 Bayesian network

A Bayesian network B is a specific type of graphical models, consisting of two parts. The first part, the structure of the Bayesian network, is a directed acyclic graph (DAG) G in which each node corresponds to a random variable. As its name suggests a DAG can be described by the following characteristics: the edges are directed and there are no cycles (i.e., starting from any given node and following the direction of the edges, there is no way to cycle back to the original node). Since under certain assumptions the directed edges can be interpreted as direct causal influences between the variables, a Bayesian network is also called a *Causal network*.

The second part is a set of conditional probability distributions Θ , that describe the conditional probability $P(X_i|Pa_i)$ of a variable X_i given its parents Pa_i in the graph G , where a parent Pa_i of X_i is defined as a node from which a directed edge points to node X_i . In this case X_i is called a child of node Pa_i .

The causal network structure in Figure 2.1 shows a sample Bayesian network for medical diagnosis [46]. It indicates, for example, that smoking can causally influence whether bronchitis is present, which in turn can causally influence whether a patient experiences dyspnoea (shortness of breath).

The link between G and Θ is defined by the Markov independencies: Each variable X_i is independent of its non-descendants, given its parents Pa_i in G . Hence Θ can be decomposed into the product form

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i|Pa_i). \quad (2.1)$$

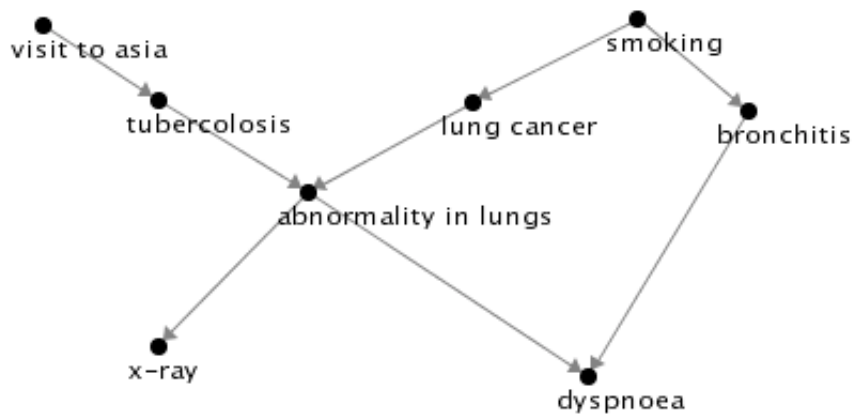


Figure 2.1: A Bayesian network for medical diagnosis; the nodes represents either diseases or other conditions and the edges represents dependencies between this nodes

2.1.1 D-separation

The complete relationship between probabilistic independence and the graphical structure of a Bayesian network is given by the concept of d-separation [38].

Definition 2.1 *Two variables a and b in a network are d-separated if for all paths between a and b there is an intermediate variable c such that either*

- *c is a node of a serial or diverging connection and its state is known.*

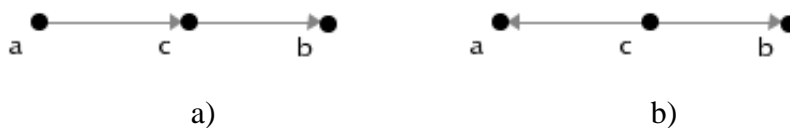


Figure 2.2: a and b are d-separated given c ; in the left structure through serial connection and in the right structure through a diverging connection.

- *c is a node of a converging connection, called collider, and neither c nor any other of its descendants have received evidence.*

It can be shown that the d-separation criterion results in the same set of conditional probability distributions as defined in Equation 2.1 [51].

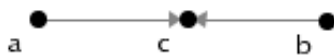


Figure 2.3: a and b are d-separated by c through a collider if c is undetermined.

2.1.2 Structure equivalence

As discussed previously, the *DAG* of a Bayesian Network describes the conditional dependence and independence relationships between a set of variables, defined by the d-separation criterion.

Theorem (Verma and Pearl, 1990) 2.1 *Two DAGs are equivalent if and only if they have the same set of edges and the same set of colliders.*

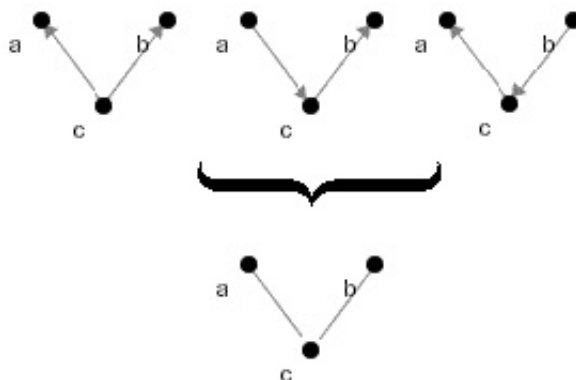


Figure 2.4: Since all three DAGs in the top row belong to the same equivalence class, they can be redrawn as a PDAG shown in the bottom row.

This implies that two DAGs, both representing the same d-separations and d-connections and hence the same probability distribution can differ in the direction of some edges. For example consider the graphs in the top row of Figure 2.4. In all three DAGs, a and b are d-separated given c. They belong to the same *equivalence class*.

Although the DAG shown in Figure 2.3 contains the same edges as the previous three DAGs it doesn't belong to their equivalence class, since it represents different d-separations and d-connections. This is because this DAG contains a collider at node c.

A set of equivalence *DAGs* can hence be redrawn as a unique *partial DAG* (*PDAG*), which consists of both undirected edges with a undefined direction and directed ones with an irreversible direction.

2.2 Structural learning

Learning Bayesian networks from data has become an increasingly active area of research [50, 23, 22, 48] and can be classified into two problems. In the first case the network structure is already known and only the parameters have to be learned from the dataset. The second task is the harder one, since besides the parameter values also the network structure has to be learned out from the dataset. This thesis focuses on the second task, since its goal is to reveal biological relevant structures out from experimental datasets. Structural learning [21, 49] is an *unsupervised learning* problem which can be stated as follows: given a *dataset* $D = \{d^1, d^2, \dots, d^N\}$ of independent observations, find the structure that best matches D . Each datapoint d^l of the N samples of dataset D is a n -dimensional vector $d^l = \{d_1^l, \dots, d_n^l\}$.

2.2.1 Scoring function

To evaluate the goodness of fit of a network with respect to the dataset, a statistically motivated *scoring function* S assigns a score $S(G)$ to the graph G . Before talking about different scoring metrics, it is necessary to outline two fundamental properties of a scoring function.

Decomposability

Since the conditional probability distribution of a Bayesian network decomposes into the product term of Equation 2.1, one might suspect that the same holds for the scoring function. In fact, if the dataset D contains neither missing nor hidden values, the score $S(G)$ can be decomposed into:

$$S(G|D) = \prod_{l=1}^N \prod_{i=1}^n P(d_i^l | Pa_i(d^l)) \quad (2.2)$$

where $Pa_i(d^l)$ denotes the set of datapoints of the parents of i in the vector d^l . This holds also if one uses the logarithmic scoring function to avoid numerical problems

$$S(G|D) = \sum_{l=1}^N \sum_{i=1}^n \log(P(d_i^l | Pa_i(d^l))) \quad (2.3)$$

As in Equation 2.1 the scoring function factorizes into terms related to the individual variable X_i dependent only on its parents, and the contribution of each variable to the total score depends only on the values of d_i^l and $Pa_i(d^l)$ in the instances of dataset D .

Score equivalence

As outlined in Section 2.2.1, due to Markov equivalence, different DAGs belong to the same equivalent class. Only in the case of prior knowledge it might be reasonable to distinguish among equivalent DAGs. Otherwise there is no reason for favoring a particular structure over another equivalent one. Due to this, the score for DAGs from the same equivalence class has to take the same value. This is called *score equivalence*.

Bayesian score

This scoring function is derived from methods of Bayesian statistics. It is proportional to the posterior probability of a network structure, given the data:

$$S(G|D) = \frac{P(D|G)P(G)}{P(D)} \quad (2.4)$$

$P(D|G)$ is the *marginal likelihood*, $P(G)$ is the prior probability of the structure and $P(D)$ is called evidence. Since $P(D)$ is constant over the various structures, it can be ignored. In addition, without prior knowledge of structures, $P(G)$ might be replaced by a noninformative prior, that is $P(G) = const.$. For domains in which the prior knowledge about the structure is present, it makes sense to construct a structure out of this prior knowledge, so that deviations from this prior structure can be penalized. In [25] for example, for learning biochemical pathways a prior structure has been constructed based on databases of known protein-protein and protein-DNA interactions. For further discussion about structural prior information see [19].

Ignoring both priors, the problem is now reduced to finding the structure with the best marginal likelihood according to the data. In other words, how likely is it that the data were generated from the structure.

$$P(D|G) = \int P(D|\Theta, G)P(\Theta|G)d\Theta \quad (2.5)$$

Given Equation 2.1 $P(D|\Theta, G)$ can be rewritten as

$$P(D|\Theta, G) = \prod_{l=1}^N \prod_{i=1}^n P(d_i^l | Pa_i(d^l), G, \Theta) \quad (2.6)$$

and $P(\Theta|G)$ is the prior density of the parameters given G .

Given a multinomial model of n variables, Heckerman et al. and Geiger and Chickering [20] proposed a set of assumption, namely *complete data, parameter*

independence and *parameter modularity* with those in combination with *Dirichlet priors* Equation 2.6 can be rewritten as

$$P(D|G) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \quad (2.7)$$

where r_i denotes the set of values which variable X_i can take on and q_i denotes the set of values which the parents of X_i can take on. $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the gamma function; for positive integers $\Gamma(x) = (x-1)!$.

N_{ijk} denotes the number of cases in dataset D in which $d_i^l = k$ and $Pa_i(d^l) = j$, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. N'_{ijk} express parameters of the Dirichlet prior distributions, and $N'_{ij} = \sum_k N'_{ijk}$. $N_{ijk} = \frac{1}{q_i r_i}$, which is a commonly used noninformative parameter prior. For more details on Dirichlet priors, see [11].

The assumption of parameter independency states that knowledge about the physical probabilities associated with a given node and parent instantiation do not influence the belief about the physical probabilities associated with another node and/or parent instantiation. The assumption of parameter modularity says that the densities of the conditional probability of a variable X_i depends only on its parents Pa_i . The assumption of complete data, says that the value of every datapoint k in the sample vector d_l is known. This assumption will be discussed in a more detailed manner in Section 2.4.

Score based on minimum description length

The second scoring function considered in this section stems from coding theory where the aim is to create a network structure that describes the data as accurately as possible with as few symbols as possible. In other words, the structure with the highest score will have a balanced contribution of the degree of accuracy with which the network represents the data, and its complexity. The description length is defined by

$$S(G, \Theta|D) = -\log P(D|\Theta, G) + \frac{\log N}{2} |\Theta|. \quad (2.8)$$

The first term is the log-likelihood of the data D given the parameters Θ and the structure G . It corresponds to the number of bits needed to describe D with the network. The second term is the description length of the network, i.e. the number of bits required to encode the network parameters. This term automatically induces the principle of *Occams razor*: a network structure with fewer edges is preferred over a network with more edges unless the log-likelihood of the more complex structure is much higher than that of the simpler one. In other words, this term penalizes the complexity of a network structure, therefore it is referred to as *penalty term*.

For multinomial distribution Equation 2.8 is

$$S(G, \Theta|D) = - \sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} + \frac{\log N}{2} |\Theta|. \quad (2.9)$$

where terms N_{ijk} and N_{ij} are the same as for the Bayesian score in Equation 2.7. Minimizing the MDL-score is equivalent to maximizing another well-known scoring function, the *Bayesian information criterion* (BIC) [44], which is exactly the negation of MDL but with a completely different origin.

2.3 Heuristic methods

As discussed in the previous section, structural learning means finding the structure with the best quality. Unfortunately this task is NP-hard hence the search-space of possible structures increases superexponentially with the number of variables [8]. For this reason, one has to resort to *heuristic search strategies* which can efficiently determine a Bayesian network close to the optimum.

2.3.1 Local search

As outlined in Section 2.2.1 the scoring function can be decomposed into a sum or a product of scoring functions restricted to each family (a variable X_i and its parents Pa_i). Each term can be defined as the *local score* of X_i . This decomposition is crucial for learning structures, since a local search procedure that changes one edge at each move can efficiently evaluate the gains made by this change.

2.3.2 Greedy search

The simplest method of heuristic search is the *greedy search*, that starts out from an initial structure and improves the structure according to a scoring function. The procedure stops if no improvement is possible, that is, there is no structure in $nbd(G)$ which has a better score than G . Then the search procedure reached an eventually local optimum. Starting out from an initial structure, e.g. an empty structure, local search proceeds through the search space of neighboring structures $nbd(G)$. The neighborhood of a current structure G is defined as the set of *DAGs* which differ only by one edge from G , e.g. cause of a single edge deletion, addition or reversion of direction. Given the current structure G in the search process, there are many neighboring structures $G' \in nbd(G)$ which can become the next intermediate structure G^* . The difference in their score $\Delta := \text{score}(G') - \text{score}(G)$

can be expressed in terms of a relative scoring function. The neighboring scoring structure which entails the largest improvement becomes the next intermediate structure G^* . Figure 2.5 sketches an algorithm for greedy search.

```
choose  $G$  somehow
FOR each  $G'$  in  $nb(G)$  DO
  Compute  $Score(G')$ 
END FOR
 $G^* := \operatorname{argmax}_{G'} score G'$ 
IF  $score(G^*) > score(G)$  THEN
   $G := G^*$ 
END IF
```

Figure 2.5: Pseudo-code for greedy hill-climbing

2.3.3 Simulated annealing

The problem of getting stuck in local optima is a big drawback of the greedy search. Various other optimization techniques, as *iterated hill-climbing* or *genetic algorithm* try to overcome this problem. One approach is the optimization with *simulated annealing*. In contrast to the greedy search, simulated annealing allows occasional uphill jumps, allowing the system to hop out of local minima and giving it a better chance of finding the global minimum. Simulated annealing exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a global optimum in an optimization problem. The algorithm is based upon that of Metropolis et al. [32], which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. The connection between this algorithm and mathematical minimization was first noted by Pincus [39], but it was Kirkpatrick et al. [27] who proposed that it form the basis of an optimization technique for combinatorial problems. Similarly to greedy search, simulated annealing employs a random search through the space of possible structures, with the distinction that it accepts not only changes that improve the structure, but also some changes that decrease it. The latter are accepted with a probability $p = e^{-\frac{\Delta}{T}}$. The pseudo-temperature T is gradually lowered throughout the algorithm from a sufficiently high starting value (i.e. a temperature where almost every proposed transition, both positive and negative, is accepted) to a "freezing" temperature, where no further changes occur.

If the temperature is decreased logarithmically, simulated annealing guarantees an optimal solution [29]. Figure 2.6 sketches the pseudocode of simulated annealing.

```

initialize  $T$ 
choose  $G$  somehow
WHILE  $T > T_{min}$  DO
  FOR  $x = 1$  to  $N$  DO
    FOR each  $G'$  in  $nbd(G)$  DO
      Compute  $\text{Score}(G')$ 
    END FOR
     $G^* := \text{argmax}_{G'} \text{score}G'$ 
    calculate  $\Delta := \text{score}(G^*) - \text{score}(G)$ 
    IF  $\Delta > 0$  or  $e^{-\frac{\Delta}{T}} > \text{random}$  THEN
       $G := G^*$ 
    END IF
  END FOR
  reduce  $T$ 
END WHILE

```

Figure 2.6: Pseudo-code for simulated annealing

2.4 Missing values

For simplicity, dataset D was assumed to be complete up to now, i.e., a value is assigned to every random variable in every case contained in the given dataset. In many situations, however, especially in biological data, one faces the problem that the available data is *incomplete*. It means that in a dataset D , which consists of a set of data vectors $D = \{d^1, d^2, \dots, d^N\}$, where each data vector consists of i components according to the number of variables, some individual components of these datavectors may be undefined $d_i^k = \text{missing}$. These undefined spaces are referred to as *missing values*. If the proportion of cases with missing data is low, the simplest solution is to throw out those cases from the dataset, given that the missing values are independent of the data. In this case the values are *missing at random* (MAR). When the proportion of missing cases is too high, it becomes important to make full use of the information which is potentially available

from the incomplete patterns. A popular methods for handling missing data is the Expectation-Maximization (EM) approach [12].

2.4.1 Expectation maximization

As its name suggests, the EM algorithm consists of two steps, *expectation* and *maximization*. Initially, the expectation step assigns some random values to the parameters in Θ . Then conditioned on G , Θ and the dataset D the *expected sufficient statistics* for missing values are computed:

$$E(N_{X_i=k, Pa_i=j}) = \sum_{l=1}^N P(X_i = k, Pa_i = j | d^l, \Theta, G). \quad (2.10)$$

When X_i and all of the values of Pa_i are observed in the sample d^l of the dataset D , the term in the sum for this sample is either 1 or 0, otherwise, when some of the variables are unobserved, the corresponding probability can be computed using an inference technique for Bayesian networks, such as variable elimination [10]. In the next step, the expected sufficient statistics are used as if they were actual sufficient statistics from a complete (imaginary) dataset D' to compute the values Θ that maximize $P(D' | \Theta, G)$:

$$\theta_{X_i=k, Pa_i=j} = \frac{E(N_{X_i=k, Pa_i=j})}{E(N_{Pa_i=j})}. \quad (2.11)$$

where $E(N_{Pa_i=j}) = \sum_{x_i} E(N_{X_i=k, Pa_i=j})$.

This two steps are repeated iteratively until a *stationary point* is reached. As shown in [12] under certain regulatory conditions, the EM algorithm converges to a local optimum.

2.4.2 Structural expectation maximization

One of the first applications of EM to learn Bayesian Networks was by Lauritzen [30]. He used it to find the parameters of a given network structure from incomplete data. The problem becomes harder when additionally to hidden values, the structure is unknown. In this case structural learning has to be applied as well as parameter learning, as described above. For this case Friedman et. al. developed an algorithm, called *structural expectation maximization (SEM)* which interleaves searching through the space of possible structures with the EM algorithm to estimate parameters. The concept is similar to that for the complete data problem, except that the score of the network is found using the expected sufficient statistics from the EM algorithm. Figure 2.7 shows the pseudocode for the sem-algorithm.

It starts from an initial structure. The structure is then passed to the EM to calculate the score. According to the schema of local search described in Section 2.3.1, a new structure is found by adding, deleting or reversing an edge and determining the new score by EM. Friedman's innovation was to save computational power by using EM parameter estimates for the current structure to evaluate candidates for the next structure, thus:

$$E(N_{X_i=k, Pa_i=j})^{G'} \cong \sum_{l=1}^N P(X_i = k, Pa_i = j | d^l, \Theta, G). \quad (2.12)$$

and to run EM again only for the structure actually chosen. If the score of the chosen structure is better than the current structure then the current structure becomes the new one. When using simulated annealing, also structures with an inferior score can become the new one (cf. Section 2.3.3). The procedure continues until no better structure can be found. For further details see [15].

```

choose an initial graph  $G$ 
FOR each  $G'$  in  $nbd(G)$  DO
  Compute expected sufficient statistics in Equation 2.12
  Compute  $Score(G')$ 
END FOR
 $G^* := \operatorname{argmax}_{G'} score G'$ 
improve parameters of  $G^*$  using EM
IF  $score(G^*) > score(G)$  THEN
   $G := G^*$ 
END IF

```

Figure 2.7: The SEM algorithm with greedy search.

2.5 Gene expression

Without going too much in details this chapter gives a brief introduction to a fundamental process in molecular biology: the *expression* of a gene in a certain protein in *eukaryotes*. For more detailed informations see [34]. In general, every cell in an organism contains the entire *genome* which is subdivided into a set of *chromosomes*. Each chromosome is a linear molecule called *DNA*, that is functionally divided into information units called *genes*. Each gene carries information for the

production of a set of proteins that perform a specialized function in the cell. Proteins are the functional building blocks of an organism.

To retrieve the information encoded in the gene, cells use the process of *gene expression*, which consists of two steps: in the step, called *transcription* the gene is copied into the form of *mRNA*, a molecule very similar to DNA. A major role in transcription plays an enzyme called *RNA-polymerase II* (PolII), which creates a new single-stranded mRNA out of the DNA sequence. A gene is now represented by a mRNA molecule that can be used as a template for the next step. In this second step the mRNA is, after being carried out of the nucleus into the cytoplasm, *translated* into the corresponding protein. A gene is said to be *expressed* in a cell if its corresponding proteins are present in the cell. During expression many mRNAs are transcribed and translated. The number of mRNAs in the cytoplasm relates to the *expression level* for that gene and can be used as a measure for the amount of proteins needed by the cell (cf. Section 2.8).

2.6 Transcriptional regulation

A fundamental question regarding gene expression is by which factors it is controlled. For most genes regulation of *transcription initiation* is the principle mechanism for controlling their expression, however gene expression can be controlled at other stages, e.g. by post-transcriptional factors. This section focuses on the regulation of transcription.

2.6.1 Structure of a gene

A gene decomposes into a *coding region* and a *regulatory region*. The coding region is the DNA-sequence, that encodes a class of protein. Nevertheless, besides coding fragments, called *exons*, it contains also fragments, called *introns*, that have no coding information. The regulatory region contains binding sites for transcription factors, which affects the process of transcription. As shown in

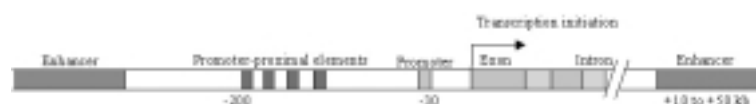


Figure 2.8: General pattern of an eukaryotic gene consisting of a coding region and a regulatory region

Figure 2.8 the regulatory region can be subdivided into several parts. The *promotor* determines the sites of transcription initiation and direct binding of RNA polymerase. *Promotor proximal elements* are control elements close to the start

site. They help regulating the transcription of a particular gene. *Enhancers* have the same function as promoter proximal regions, but they are found at a greater distance from the transcription site, either upstream or downstream of the gene or within an intron. Promoter proximal regions and enhancers are also called *cis-regulatory elements*.

2.6.2 The transcription initiation complex

The promoter site binds Pol II and its accessory factors, called *general transcription factors*, and directs the PolII to begin transcribing at the correct start site. The single-stranded DNA sequence is then transcribed into a corresponding mRNA sequence. This complex of different molecules is necessary for the catalytic process of transcription.

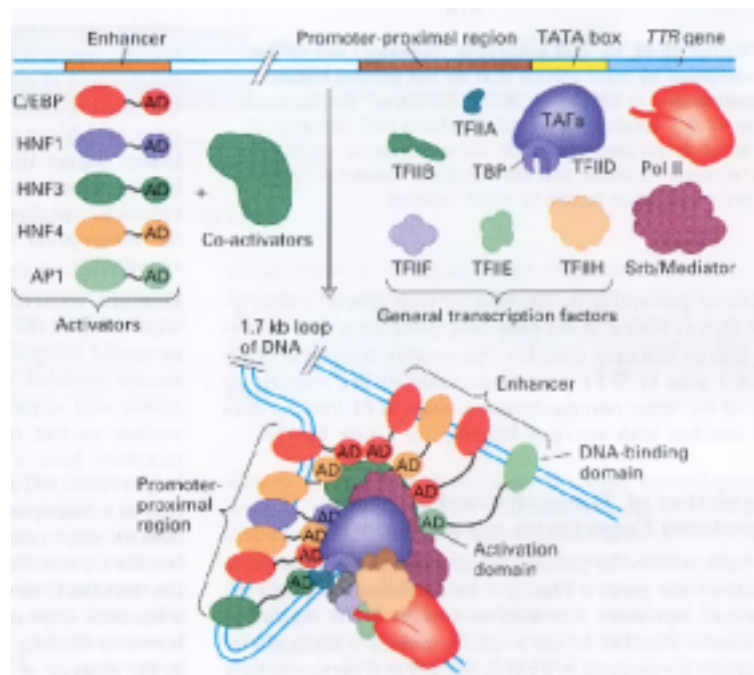


Figure 2.9: A typical transcription complex consisting of the transcription initiation complex and several regulatory proteins. Figure taken from [34].

2.6.3 Regulatory proteins

Additionally, transcription is regulated by binding of sequence-specific DNA-binding proteins, so called *regulatory proteins*, to regulatory promoters and en-

hancers as shown in Figure 2.9. By interacting with the general transcription complex they can either activate or suppress the transcription.

Activators

Activators are generally modular proteins containing a single *DNA-binding domain* and one or few *activation domains*. The binding domain targets the activator to a specific site, whereas the activation domain interacts with the transcription initiation complex. The different domains frequently are linked through flexible polypeptide regions. This may allow activation domains in different activators to interact even when their DNA-binding domains are bound to sites separated by tens of bp.

Suppressors

Suppressors have the same structure as activators, however their mechanisms are less well understood than activation mechanisms. In general, transcriptional suppressors can be divided into 3 categories: First suppression by inactivation of an activator. Second, by inhibition of the formation of the preinitiation complex, also called *global suppressors*. This type of suppressors may play an integral role in the transcriptional activation process, in that some activators may stimulate the formation of a preinitiation complex by displacing a global suppressor. Thirdly by inactivation of an activator as well as of the transcription initiation complex of a given gene.

2.6.4 Tumor suppressor genes

The lack, or dysfunction of such regulatory proteins can have serious consequences especially in context with tumor-specific abnormalities. The *WT1-protein*, for example, is a suppressor that is expressed preferentially in the developing kidney. Children who inherit mutations in both the maternal and paternal *WT1-gene*, so that they produce no functional WT1-protein, invariably develop kidney tumor early in life. In other words inactivation of the WT1 gene has been correlated with the incidence of Wilms tumor, a pediatric kidney cancer. The WT1 protein binds to the control region of a gene that encodes a transcription activator called *EGR-1*. Thus leads to its classification as a *tumor suppressor gene*.

Another tumor suppressor gene is *p53*, a regulator of DNA damage. Individuals who inherit only one functional copy of p53 are also predisposed to cancer. The p53-protein regulates the expression of another protein: *p21*. This protein prevents cells from prematurely entering S phase, especially if the DNA has lesion. Thus DNA damage will stimulate the production of p53 to arrest the progression

of the cell cycle until the lesion is repaired. If the DNA damage is too severe, the p53 protein directs the cell toward apoptosis or cell death. If the cell lacks functional p53 protein and the cell is able to survive the accumulated gene damage, it progresses towards an increasing malignancy. They continue dividing in an uncontrolled way forming aggressively growing tumor tissues.

Suppressor gene	Tumor type
BRCA1	Breast, ovary
BRCA2	Breast (both sexes)
p53	Breast, sarcoma
APC	Colon
MTS1	Skin, pancreas
CDK4	Skin
NF-1	Brain, other
NF-2	Brain, other
WT1	Wilms' tumor
VHL	Kidney, other
RB	Retinoblastoma, sarcoma, other

Figure 2.10: Inherited mutations in these genes confer a very high cancer risk.

2.7 Cell cycle regulation

Cell division is divided into 4 major phases: G1, S, G2 and M, which are the traditional subdivisions of the *cell cycle*. The most significant phases are the S (synthesis) phase, during which the chromosomes are replicated, and the M (cytokinesis) phase, during which the cell divides into two *daughter cells*. Since cell division is a very critical mechanism cells are controlled on three different checkpoints throughout the cell cycle, that are, the G1 checkpoint in late G1, the G2 checkpoint between G2 and M and M checkpoint in late M phase.

The cell cycle checkpoints are controlled by *cyclin dependent kinases* (CDK) which are composed of two proteins - a *cyclin* and a *kinase*. As described above, any mutation that removes or otherwise modifies a checkpoint inhibitor such as *p53* or removes or modifies a transcription inhibitor such as *Rb* will lead to a loss of cell cycle regulation and lead to cancer.

In yeast *Saccharomyces cerevisiae*, until now one CDK protein, namely *cdc28*, and 10 different cyclins, 4 *clns* and 6 *clbs*, have been identified. One of these cyclins, namely *CLN2*, is encoded in the *YPL256C* gene, which will be examined in Section 3.2.3. The identification of such cell cycle regulating genes becomes

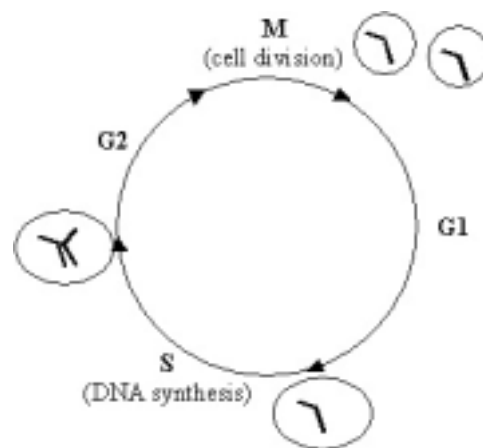


Figure 2.11: The cellcycle of an eukaryotic cell

more and more popular. Many experiments examine the expression level of thousand of genes to reveal those which oscillate with the cell cycle phases. Due to this a huge amount of datasets is available to analyze them in a statistical manner.

2.8 DNA microarrays

Microarrays are one of the latest breakthroughs in experimental molecular biology. They allow monitoring of gene expression for tens of thousands of genes in parallel. Although mRNA is not the ultimate product of gene expression, information about the transcription level is necessary to understand gene regulatory networks. The correlation between the mRNA and the protein concentration in the cell may not be straightforward, nevertheless the absence of mRNA in a cell is likely to associate with a not very high level of the respective protein. For a more detailed discussion see [5].

As developed at Stanford University [43], a microarray is a glass slide, onto which single-stranded DNA molecules are attached at fixed locations, called *spots*. Such a microarray may consists of thousands of spots, each related to a single gene. The central principle of the microarray-technique is the selective binding of complementary single-stranded nucleic acid sequences, called *hybridization*.

One of the most popular experimental methods is to compare the mRNA levels across two cell cultures (e.g a cancer cell and a healthy cell as control).

The first step is to extract purified mRNA from both cell types. mRNA can be captured by using complementary strands (*oligodeoxythymidine*), which binds to the mRNA, since every mRNA sequence contains 150-200 *adenines*, called *poly-A-tail*. Hence the extracted mRNA's are unstable, they are reversetranscribed to

complementary DNA's, called *cDNA's*, which are more stable. In order to distinguish *cDNA's* from different cell types, fluorescent labeling molecules of different colour (usually red and green) are used to stain each sample, e.g. by using a red dye for the *cDNA's* from the cancer cell and a green dye for that from the healthy one.

After labeling the *cDNA* molecules, both extracts are washed over the microarray where they bind selectively to their complementary DNA-strands in the spots according to the principle of hybridization described above.

The last step of the experiments measures the hybridization level of each spot on the array. The scanning is done by an excitation laser and an emission detector. If the mRNA from the cancer cell is in abundance, the spot will be red, if the mRNA from the healthy cell is in abundance, it will be green. If mRNA from both cells bind equally, the spot will be yellow, while if neither binds, it will be no fluorescence and the spot will be black.

The resulting image gives quantitative, but strongly noisy expression measurements, which are relative by nature, hence the expression levels of the same gene in two different samples is compared.

2.9 Modelling gene regulatory mechanisms

As discussed in Section 2.1 a Bayesian network is a representation of a joint probability distribution over a set of variables. This representation consists of a network structure, and a parametrization which describes a conditional distribution for each variable. A Bayesian network is suitable to detect a net of mutual dependencies. In genetic regulatory networks, regulatory relationships are not yet well-known, but as shown above, highly relevant for understanding diseases and consequently guiding drug therapy. The data pool consists of noisy gene expression data, from which we wish to infer regulatory pathways. Applying this to the biological system described above, each variable represents a gene whereas an edge describes a causal relationship, e.g. a regulation mechanism, between two genes. In Figure 2.12 for example, gene *d* and gene *a* are related in some joint biological interaction or process, namely gene *a* influences gene *d*. The same assumption holds for gene *b* and gene *c*, where gene *b* is influenced by gene *c*. In a biological sense gene *a* and gene *c* code for regulatory proteins which influence the transcriptional process of gene *d* and gene *b*. The information about the type of transcription regulation due to gene *a* resp. gene *c* is encoded in the joint probability table of their children, shown in Figure 2.13.

Every gene has three discrete states, namely underexpressed (-1), normal (0) and overexpressed (+1).

If gene *c* is underexpressed, the probability of gene *b* to become underexpressed

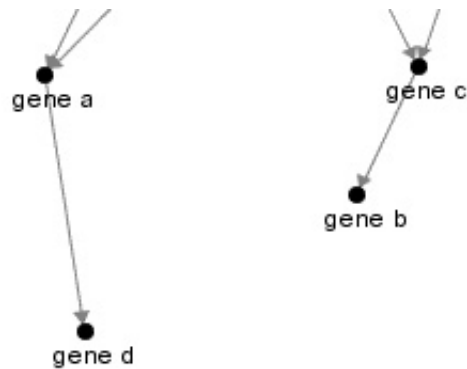


Figure 2.12: A regulatory network modelled by a Bayesian network

		gene a					gene c		
		-1	0	1			-1	0	1
gene d	-1	0	0	0.8	gene b	-1	0.8	0	0
	0	0.2	0.4	0.2		0	0.2	0.3	0.1
	1	0.8	0.6	0		1	0	0.7	0.9

Figure 2.13: The conditional probability tables of variables gene d and gene b.

is high, namely $P(\text{gene } b = -1 | \text{gene } c = -1) = 0.8$, and if gene c becomes overexpressed gene b will be overexpressed with a probability of $P(\text{gene } b = 1 | \text{gene } c = 1) = 0.9$. This leads to the assumption that gene c acts as an activator on gene b . Gene a , however, seems to suppress the transcription of gene d since in the underexpressed state of gene a gene d is overexpressed with a probability of $P(\text{gene } d = 1 | \text{gene } a = -1) = 0.8$, whereas in the overexpressed state of gene a the probability of gene d to be overexpressed is only $P(\text{gene } d = 1 | \text{gene } a = 1) = 0$.

2.10 Implementation

The framework has been developed in Java (Jdk 1.3) and tested on various computer platforms as Sun Solaris, Linux and Windows.

Chapter 3

Results

This chapter is divided into two parts. The first part deals with benchmarking structural learning on toy datasets, sampled from well known Bayesian networks. The second part applies structural learning to a microarray dataset, to reveal regulatory network structures, which are then compared with results from other machine learning techniques as well with knowledge from molecular biology.

3.1 Benchmark experiments

As said above, this section analyzes two general problems of structural learning namely the difference between structural learning with greedy search and simulated annealing, and on the other hand the effect of the sample size of the dataset on the results of structural learning. For this several experiments are applied to datasets carried out from two different benchmark Bayesian networks described in Appendix 5.

3.1.1 Greedy search vs. simulated annealing

This section analyzes the two heuristic techniques, greedy search and simulated annealing, in combination with structural learning. For this, both algorithms described in Section 2.3 are applied on a dataset of 10000 samples, drawn from the *asia network*, which consists of 8 nodes with discrete values and 8 edges. For a more detailed description see Appendix 5.1.

Since, as discussed in Section 2.2.1, the score of a structure is a measure of its quality, in terms of describing the statistics, the resulting score of a founded structure is also a measure of the quality of the search procedure, namely how good is the optimum found by the algorithm in terms of the score or how near is the found optimum to the global optimum, which is known for the toy datasets. In

the following experiments the scores of the found structures are compared with those of the best structure. It is important to say, that for the data sample used here the structure with the best score, differs from the original network shown in Figure 5.1 by one edge, namely in that way that the edge between variable A and T is absent. This is because the finite dataset sampled from the known Bayesian network may not represent all its probability distributions.

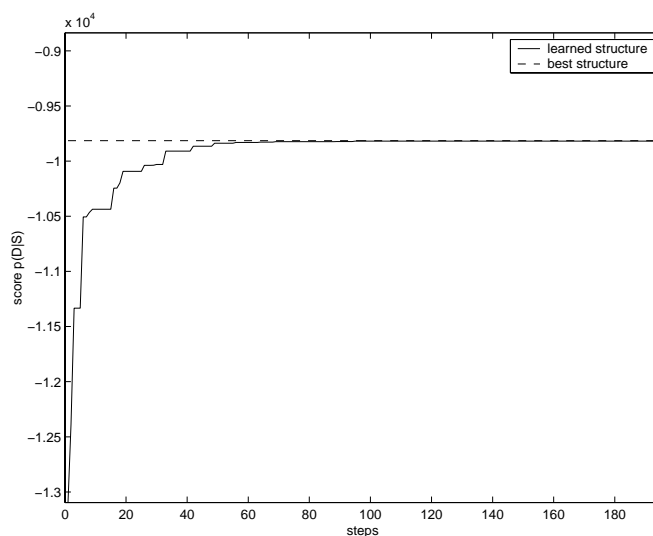


Figure 3.1: Curve of the scoring function during structural search with greedy hillclimbing in. It corresponds to the second run with greedy search in Figure 3.4. After app. 80 steps it gets stopped in a local optimum, which is near to the global one.

Greedy search starts from the empty graph and walks through the space of possible structures. At each step it picks out randomly two nodes, and from the set of possible neighboring structures, it selects the one with the highest score. If its score is higher than the actual score, the neighboring structure becomes the new one, otherwise the actual structure remains the same. If an optimum is reached, e.g. when in 100 steps no new structure is accepted, the search procedure stops. Figure 3.1 shows the course of the scoring function during greedy search: after a set of uphill-steps, search-procedure stops, hence no more changes are possible. As greedy search, simulated annealing starts from the empty structure with a high initial pseudo-temperature T_0 . Each pseudo-temperature T_n is hold until the number of proofed structural changes is equal to the number of variables in the structure. Through this the algorithm depends on the size of the search-space, which is important to get good sampling statistics for the current pseudo-temperature.

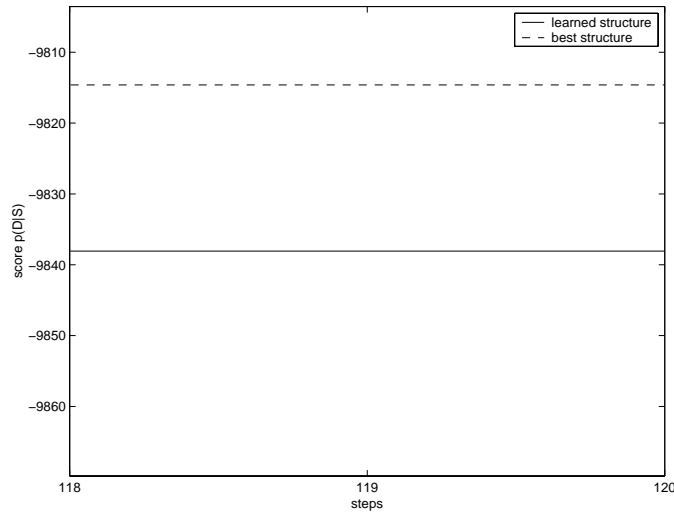


Figure 3.2: Curved of the scoring function of the last steps of greedy search zoomed out from Figure 3.1. It shows well that the structural search procedure get stuck in a local optimum.

After that the pseudo-temperature T_n is decreased to T_{n+1} as follows

$$T_{n+1} = \rho T_n \quad \rho < 1 \quad (3.1)$$

where ρ is referred to as *cooling rate*, and the above steps are repeated at the new pseudo-temperature T_{n+1} .

The major difficulty in implementation of the algorithm is that the performance of simulated annealing depends on the annealing schedule that is, the choice of initial pseudo-temperature T_0 , how many iterations are performed at each pseudo-temperature, and how much the pseudo-temperature is decremented according to the cooling rate ρ .

The curve shown in Figure 3.3 is typical for simulated annealing. In the first steps, due to a high initial pseudo-temperature, almost all structural changes are accepted, regardless of the scoring function. This is characterized as a noise-like signal at the beginning of the search, which then becomes less fluctuating in form of a deterministic trend due to the decreasing probability of accepting inferiorer structures.

Error measure

To compare the efficiency of the different heuristics, each algorithm was applied ten times to the same dataset. For each resulting structure, the corresponding scoring function was compared with the best scoring function as discussed above.

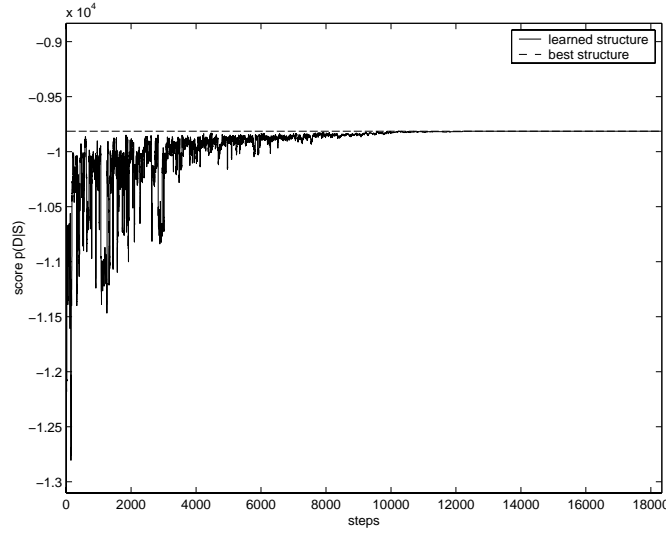


Figure 3.3: Curve of the scoring function during structural search with simulated annealing. It corresponds to the second run with simulated annealing and a cooling rate of 0.9 in Figure 3.4. It takes more steps than the greedy one, but in contrast to them, it converges to the global optimum.

As a measure for how near the determined structure is to the best structure it was used the *Kullback Leibler distance*, KL-distance, between two probability mass functions $p(x)$ and $q(x)$, where $p(x)$ denotes the empiric mass function and $q(x)$ the true one. The KL-distance is defined as follows:

$$D(p||q) = \sum_x p(x) \frac{p(x)}{q(x)} \quad (3.2)$$

The KL-distance is always non-negative; it is 0 if and only if $p \equiv q$, and it can be understand as a distance measure between two distributions (Kullback and Leibler 1951). Applying this measure to the current problem, $p(i)$ denotes the score of the learned structure and $q(i)$ that of the best structure, where both scoring functions decompose into i localscores, as discussed in section 2.2.1.

The distance of each found structure and the average distance over several runs of the various search-procedures, is shown in Figure 3.4. As one can see, greedy search never finds the optimal structure, it always gets stuck in a local optimum. But also with simulated annealing and a cooling rate of 0.9, the global optimum is reached only 4 times, whereas in the other runs it gets stuck in a local optimum like greedy search. As it is clear that only with a sufficiently small cooling rate the propability of finding the global optimum is 1 [29], simulated annealing was slowed down by setting the cooling rate up to 0.95, and 0.99. As one expect, the smaller the cooling rate the better the results. Figure 3.4 shows, that with a cool-

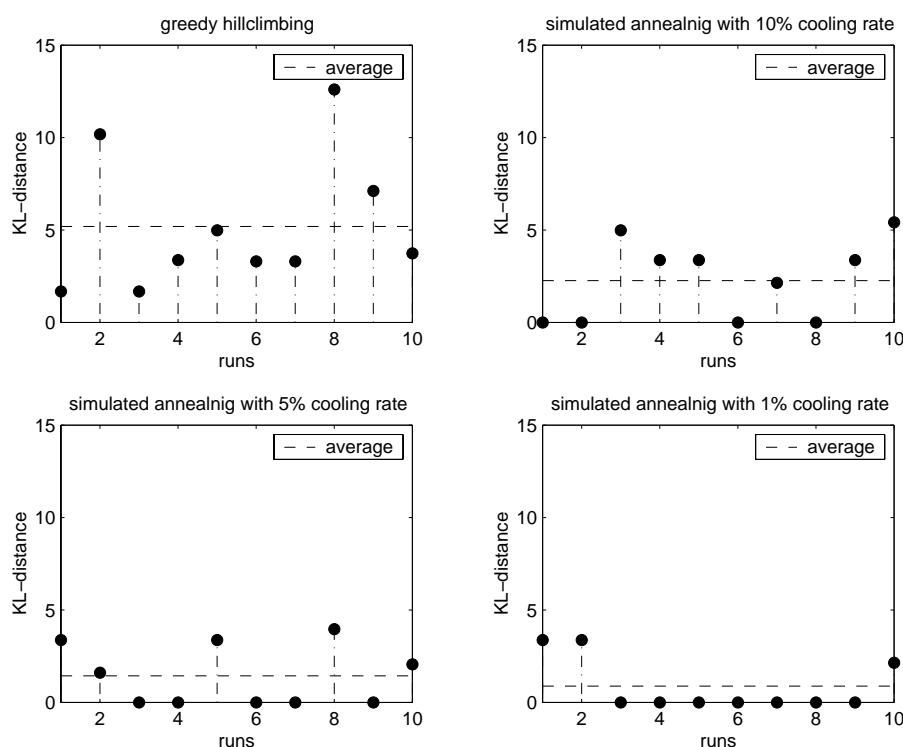


Figure 3.4: Error bars of the different search procedures over 10 runs; the error is defined by the KL-distance measure between the score of the best structure and the score of the estimated structure. The lower the distance the better the reached optimum. At first glance, one can see that simulated annealing finds better optima than greedy search, and that the result of simulated annealing can be improved by slowing down the cooling procedure.

ing rate of 0.99 even 7 times the global optimum was reached.

Nevertheless, it may be mention that there is a cooling procedure that guarantees finding the lobal optimum but it is impractical, since computation time is limited. This requires to make a compromise between a good optimization and a tractable computational time.

Another method to analyze the efficiency of a learning procedure is to compare the learned PDAG with the best PDAG by simply counting the number of erroneously present edges (false positive) and erroneously absent edges (false negative).The resulting error bars are shown in Figure 3.5. Although both measures judge different properties of the estimate, they yield similar results. This justifies, that the scoring function is a good measure to evaluate a network structure. In addition, the high number of wrong edges of the greedy search (up to 8 edges wrong in a node network), shown in Figure 3.5, demonstrates the drastical improvement

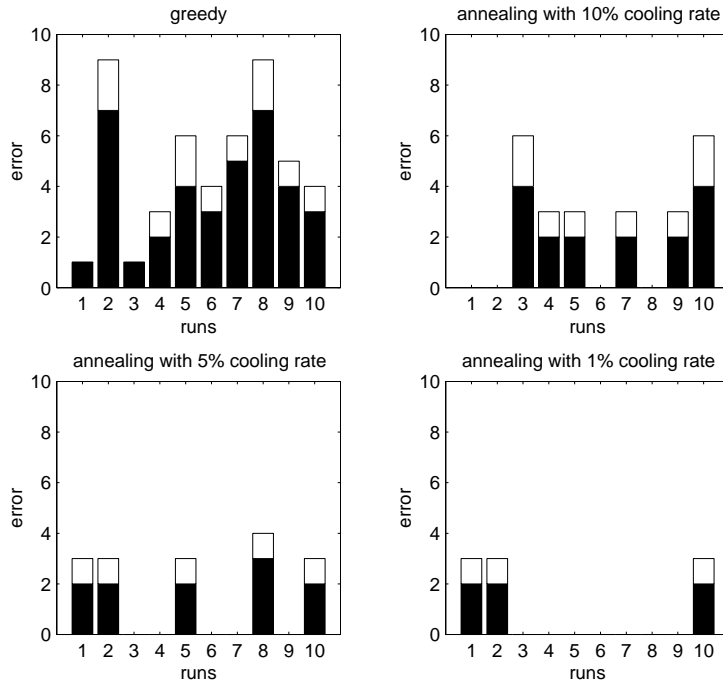


Figure 3.5: Error of structures learned with different search procedures over 10 runs; each error bar consists of the error induced by false-positive edges (black bars) and the error induces by false-negative edges (white bars) regarding the best structure

introduced by simulated annealing.

Feature PDAG

To visualize, the efficiency of both algorithms, the set of resulting *PDAGs* g (cf. Section 2.1.2), is combined to a *feature PDAG* (fPDAG), where the *feature* f of an edge is given by:

$$P(f|D) = \sum_{G \in \mathcal{G}} P(G|D) f(G). \quad (3.3)$$

The first term, $f(G)$, is 1 if G contains feature f , otherwise it is 0. The second term, $P(G|D)$, is the marginal likelihood of the structure G given the data. The confidence of a certain feature can then be visualized by selecting the line width of the edges according to their probability.

Figure 3.6 shows the resulting fPDAGs for the simulations shown in the previous section. At first sight, it can be noticed that all fPDAGs are similar to the best DAG shown in a); only the fPDAGs of greedy search and that of the fastest Simulated annealing have spurious edges. As shown already in the previous Section, by

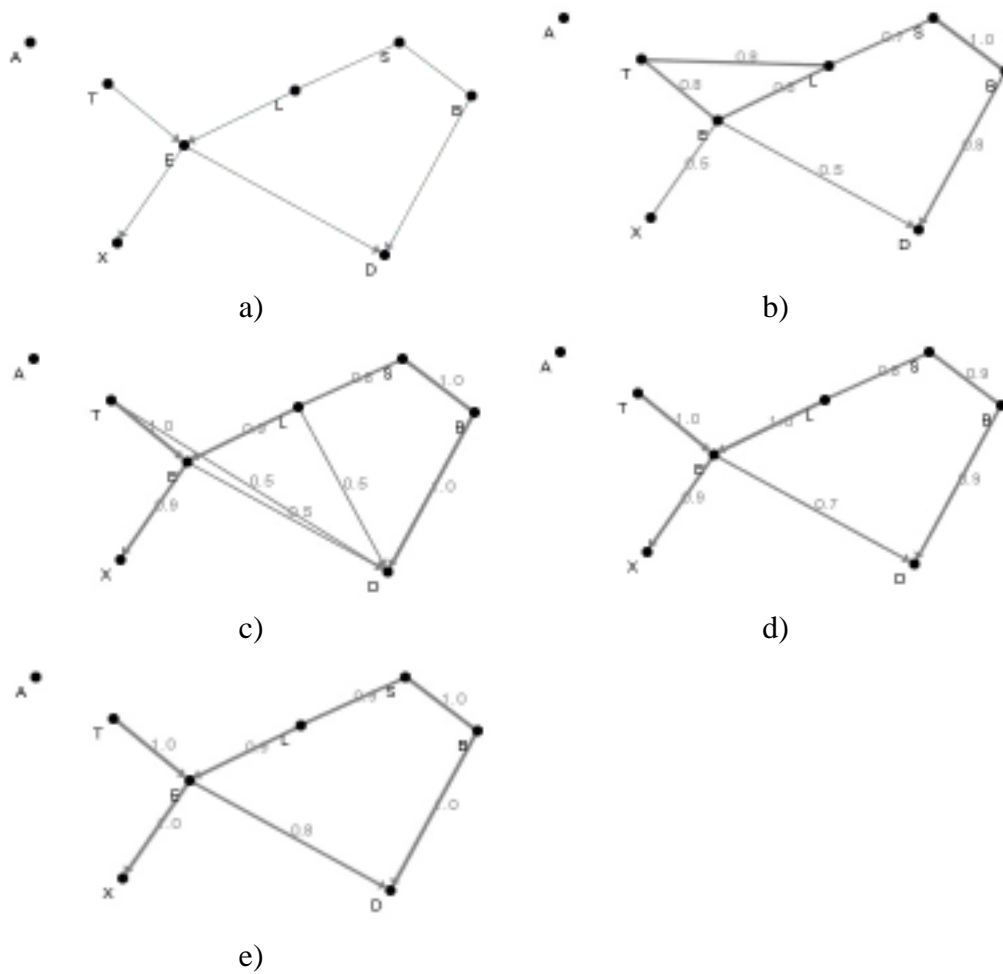


Figure 3.6: a) shows the PDAG with the best score. The other graphs are the resulting fPDAGs from different search algorithms over 10 runs; in b) the fPDAG with greedy search in c) with simulated annealing and $\rho = 0.9$ in d) with a cooling rate of $\rho = 0.95$ and in e) with $\rho = 0.99$. Each edge is characterized by its confidence level, which additionally is visualized by the line width of the edge.

decreasing the cooling rate the fPDAG becomes more and more confident. With a cooling rate of 0.99 the structure of the resulting fPDAG is identical to the best structure, and all features have a confidence level near to 1.

The experiments shown however, that neither greedy search nor simulated annealing guarantee to find the optimal structure. With both methods good optima are found, nevertheless simulated annealing finds better ones than greedy hillclimbing on average.

3.1.2 Various sample sizes

Since current Microarray datasets usually are sparse in sample-size but very large in the number of variables, namely $n = 6177$ genes and $N = 73$ samples, it is important to take a closer look at large network structures learned out from sparse dataset.

Therefore, structural learning combined with simulated annealing and a cooling rate of 0.9, was applied to datasets with different sample sizes, ranging from 30.000 down to 10. To simulate data sparseness on a network with many variables datasets are sampled from the probability distributions of the alarm network. This network is one of the biggest benchmark networks in terms of dimensionality containing 37 variables with discrete values. For a more detailed description see Appendix 5.2.

To get a measure of the divergence of a learned structure from the original structure, its resulting PDAG is compared with the original PDAG as previously. In each learned structure the number of erroneously present edges (false positives) and the number of erroneously absent edges (false negatives) are counted. It is important to point out that the learned structures are compared with the original structure, which may not be the structure with the highest score respect to the given dataset, since finite datasets are not able to exhaustively represent the probability distribution given by the Bayesian network from which they are sampled out.

Figure 3.7 shows the error of structures learned from different datasets, induced by erroneously absent and present edges. With a dataset size larger than 10000 samples the average of false negative or false positive edges is low with respect to structures learned from datasets with lower sample sizes. With decreasing the sample size the error induced by false positive and false negative edges increases. Especially the number of false negative edges increases fast with decreasing sample size. That's because with decreasing number of samples the number of edges in the learned structures generally decreases as well, since the dependencies encoded in the dataset become statistically insignificant as Figure 3.8 shows.

This leads to the final conclusion, that a structure learned from a sparse dataset becomes also sparse and that the dependencies encoded by the structure must be

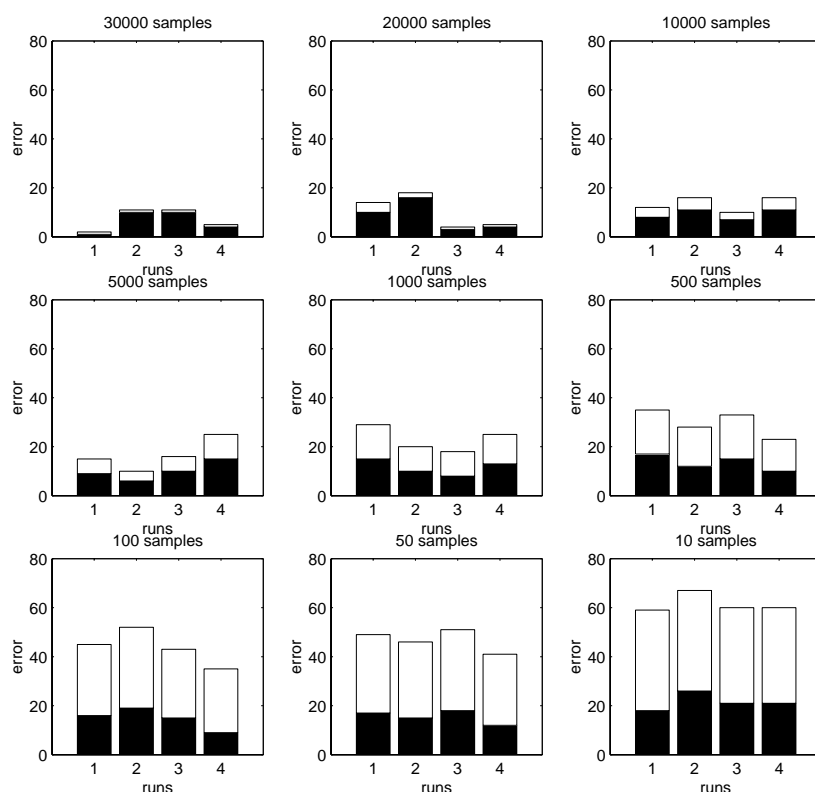


Figure 3.7: Error of structures learned out from different sample sizes over 4 runs; each error bar consists of the error induced by false-positive edges (black bars) and the error induced by false-negative edges (white bars) regarding the true structure

very strong to be still detectable. However the dependencies that are detected in the structure are relatively reliable even if the dataset is small.

3.2 Microarray experiments

After several experiments with benchmark datasets, the following experiments are applied on a microarray dataset, to reveal regulatory mechanisms.

3.2.1 The microarray experiment by Spellman et al.

The dataset, which will be used in this section, consists of the set of microarray images resulting from different experiments made by Spellman et al. [45]. All data are publicly available from the homepage of Stanford University [33]. The

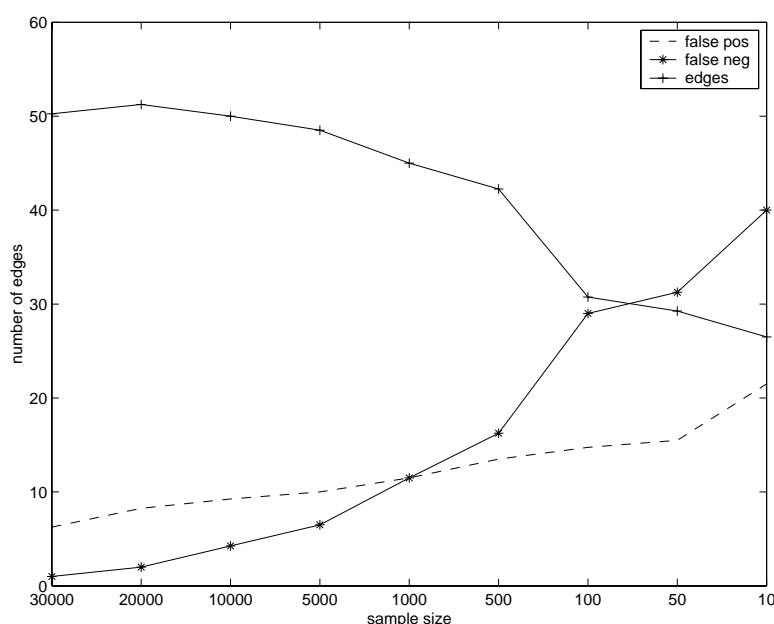


Figure 3.8: The number of false positive, false negative and significant edges of the learned structures out of different sample sizes.

goal of these experiments was to identify cell-cycle-regulated genes of the yeast *saccharomyces cerevisiae*. For this DNA microarrays were used to analyze mRNA levels in cell cultures at different timepoints during the cell cycle.

Spellman et al. treated the *S. cerevisiae* cell cycle by four different methods, namely by α -factor-, *cdc15*- and *cdc28*- based blocking, and *elutriation*. Since in each methods different temperatures were used, the cell cycle occurred at different rates.

Figure 3.9 shows the time course of the relative mRNA level for ORF *YKL164C*. In the α -experiment the cell culture passes two cell cycle periods in the *cdc15*-experiment 3 periods, in the *cdc28*-experiment 2 periods and in the *elutriation*-experiment one period.

Additionally the experiments differ in the interval in which mRNA levels were observed. In the α -experiment a microarray image every 7 minutes was made, in the *cdc15*-experiment and in the *cdc28*-experiment every 10 minutes, and in the *elutriation*-experiment every 30 minutes.

By a Fourier analysis and a correlation measure each gene was tested for periodicity and assigned with a cell cycle-dependent clustering score (CDC-score). Genes were ranked by their aggregate CDC-scores, and the list was examined to determine a threshold that was exceeded by 91% of known cell cycle-regulated genes.

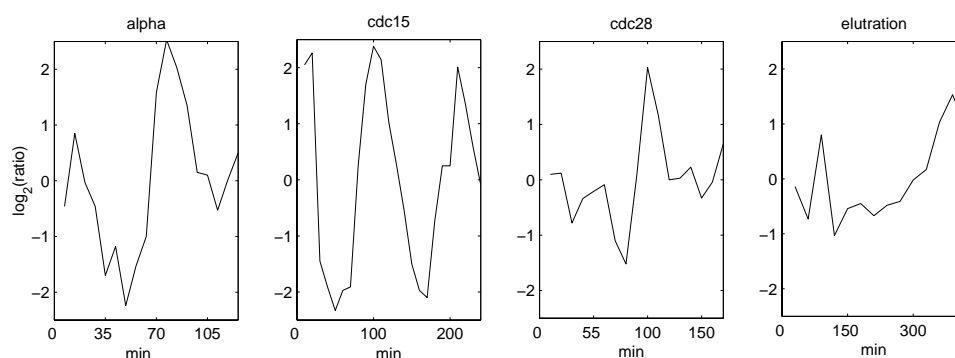


Figure 3.9: The $\log_2(\text{ratio})$ of the relative mRNA level of gene YKL164C during four different experiments with different rates of cell-cycles.

So from the 6177 ORFs¹ tested on the microarray 800 ORFs were identified as being periodically regulated according to their CDC-scores.

Additionally, Spellman et al. classified these 800 ORFs first by analyzing their pattern of expression, which is referred to as “phasing” and secondly by a cluster algorithm of Eisen et al. [14]. This algorithm searches through all the data to find the pairs of genes that behave most similarly in each experiment and then progressively adds other genes to the initial pair to form clusters of apparently coregulated genes.

The results from this experiment are used to examine the results presented in this work.

3.2.2 The microarray-dataset

Although the number of variables was reduced from 6177 to 800 cell-cycle relevant genes, the number of samples is much smaller than the number of variables, namely 73 samples, which corresponds to a sample-variable ratio of less than 0.1. As already discussed in the previous section, this leads to the assumption that only a sparse structure can be learned reliably given such a low sample-variable ratio. Additionally, the dataset contains about 10% of missing values, which require the use of algorithms which are able to handle missing datasets, as described in section 2.4.

The resulting dataset is a matrix X , where each row corresponds to a gene and n columns, where each column corresponds to an experiment. Each value x_{ij}

¹An open reading frame (ORF) is an interval on the DNA which potentially codes for protein. Its structure is similar to the general pattern of a gene described in section 2.8. This term is often used when, after the sequence of a DNA fragment has been determined, the function of the encoded protein is not known.

contains a numerical representation of the *expression level*, that is

$$x_{ij} = \log_2 \frac{Cy5_{ij}}{Cy3_{ij}} \quad (3.4)$$

where $Cy5_{ij}$ denotes the intensity of fluorescence of the red labeled sample of gene i in the experiment j and $Cy3_{ij}$ denotes the intensity of fluorescence of the green labeled sample, the control sample. Mention that $x_{ij} > 0$, gene i is overexpressed and if $x_{ij} < 0$, gene i is underexpressed relative to the control sample. If the mRNA levels of gene i are the same in both cell cultures $x_{ij} = 0$. The value of each spot was additionally normalized by the average over one experiment.

To apply the multinomial model the expression values are discretized into 3 values: under-expressed (-1), normal (0) and over-expressed (+1). The threshold

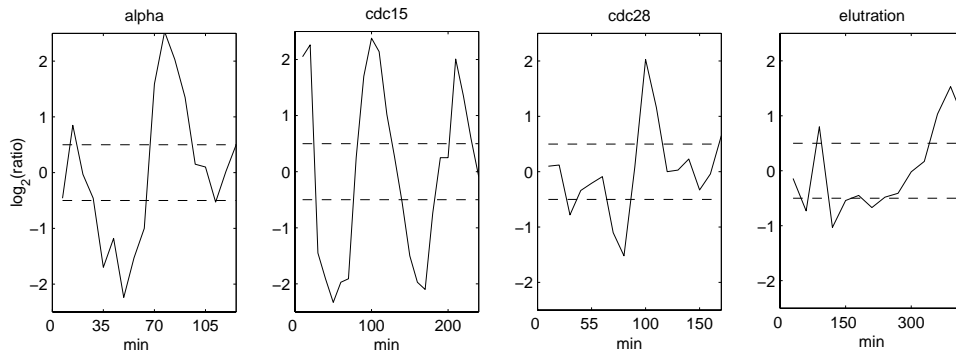


Figure 3.10: The $\log_2(\text{ratio})$ of the relative mRNA level of gene YKL164C during four different experiments and the thresholds for discretizing the mRNA level.

value is set to 0.5 in logarithmic scale (base 2), as Friedman et al. [16], namely that values of x_{ij} with a ratio lower than $2^{-0.5}$ are considered as under-expressed and values higher than $2^{0.5}$ are considered over-expressed.

3.2.3 Learned subnetworks

Due to computational costs and limited time, the search-algorithm was applied on small subnetworks of about 10 or more variables. Some runs were done with all 800 genes, but the results showed that without restricting the search space it is impossible to find out a good reasonable structure. Therefore in the future there should be find ways for restricting the search space, considering statistical as well as biological aspects, to be capable of analyzing the complete dataset.

The variables in learned subnetworks correspond to selected ORFs with the three discrete expression levels as values, plus an additional variable, which represents the phase of the cell-cycle. This node has the same properties as the other ones,

except that it is a root node, which means that it can be only a parent node and never a child node. The values of these nodes are the 4 phases of the cell cycle, G1, S, G2 and M. By inserting this node the conditional dependencies encoded by the network include also the temporal aspect of the microarray experiments made by Spellman et al. [45]. From all subnetwork-datasets 20 networks were learned each with another random seed and then combined to a feature graph, as discussed in Section 3.3, for minimizing the error made by heuristic search.

The YPL256C-subnetwork

The YPL256C subnetwork consists of 12 nodes, with the 3 expression levels (-1, 0, +1) as discrete values, representing different ORFs and 1 node which represents the phases of the cell-cycle. Since this subnetwork was already published by Friedman et al. [16], it was chosen to be able to compare the outcome of the network to their results. Out of this dataset 20 PDAGs were learned, which were summarized to the feature PDAG, shown in figure 3.11. The main node of this network represents the ORF *YPL256C*, which has a central role in cell cycle regulation. It is known to be essential for the control of the cell cycle at the G1/S transition.

As one can see, confidence in the resulting fPDAG is not too high, since the variance of the learned structures is considerable. Nevertheless there are two features with a confidence level of 0.8.

One is the directed edge from *YPL256C* to *YIL066C*, which encodes a causal dependency relationship between these two ORFs. Both appear in the same cluster of Spellman et al. since their expression is correlated. It also suits biological knowledge: as discussed above, *YPL256C* encodes a *G1 – cyclin* involved in regulation of the cell cycle. On the other hand *YIL066C* plays a role in DNA replication, which appears in the S-phase. So a causal dependence of *YIL066C* from *YPL256C* seems biologically reasonable.

The other high-confident feature is the directed edge from *YMR001C* to *YLR131C*. *YMR001C* encodes a protein which is involved in regulation of DNA replication. *YLR131C* encodes a transcription factor that activates transcription of genes expressed in the G1 phase of the cell cycle. The fact that *YLR131C* is transcribed in G2 phase and *YMR001C* is active in the S phase, confirms the presence of that feature, which encodes a causal dependence of *YLR131C* from *YMR001C*. So *YMR001C* is likely to regulate *YLR131C*.

Another ORF influenced by *YMR001C* is *YDR146C*, which encodes a transcription factor that activates transcription of genes expressed at the M/G1 boundary and in G1 phase of the cell cycle. *YDR146C* itself influences *YHR023W*, which encodes a protein that plays a non-essential role in cytokinesis, in M phase. An unexpected fact is, that ORF *YGR108W*, which encodes a B-type cyclin that

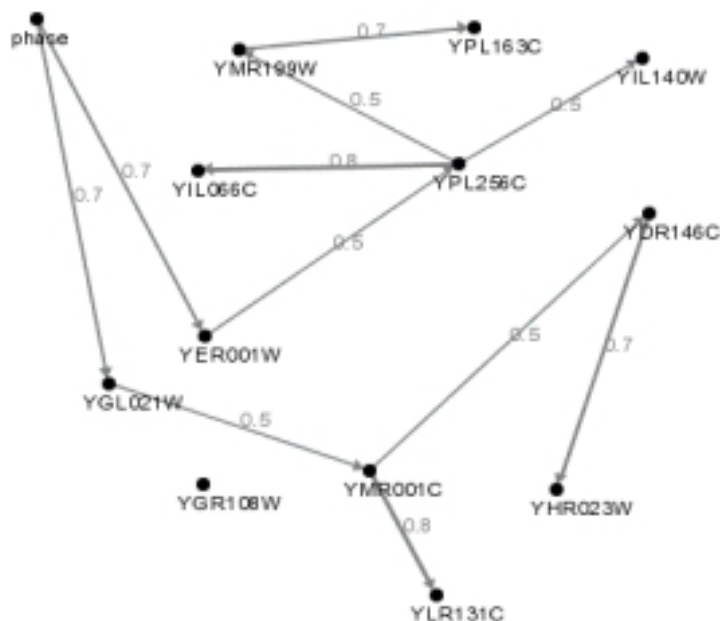


Figure 3.11: The resulting feature PDAG learned from the *YPL256C*-subnetwork; it consists of 12 ORFs and the phase node. *YPL256C* is the central ORF by influencing 3 other ORFs.

activates ORF *YBR160W*, which doesn't belong to the subnetwork, to promote the transition from G2 to M phase of the cell cycle, has neither parents nor children, although in Spellman et al. it belongs to the same cluster as the other cyclins. This is because the dataset is too sparse for an edge between *YGR108W* and another node. ORF *YPL256C* plays a major role in cell cycle regulation, namely it controls the state of the cell at the late G1 phase. Also in the learned network *YPL256C* plays a major role by influencing 3 other ORFs, namely *YIL066C*, *YMR199C* and *YIL140W*. These relationships are biologically reasonable since *YIL140W* belongs to the same cluster and *YMR199W* also encodes a cyclin involved in regulation of the cell cycle at the G1/S-transition. This example shows that the causalities expressed by the network appear reasonable in a biological

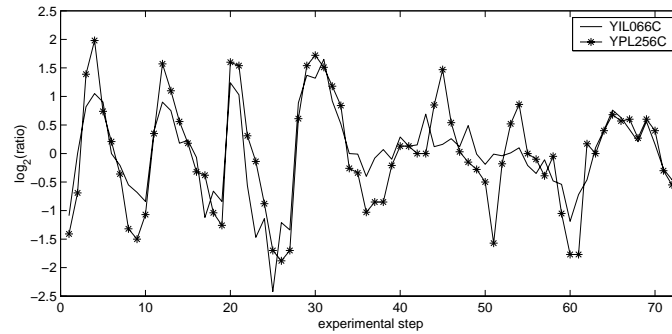


Figure 3.12: The time courses of the expression levels of ORF *YIL066C* and *YPL256C*. Both are strongly correlated, but no obvious phase shift is present, which would confirm the directed causal relationship found by the Bayesian network.

sense. This leads to the assumption that learning of Bayesian networks is suitable for revealing also new unknown relationships between genes.

It should be annotated, that without phase node, the learned network structure has the same edges as in Figure 3.11 but all without an obvious direction which can be explained in two ways. First by the fact, that without phase node the network structure lacks of any colliders which leads to an completely undirected graph as shown in Section 2.1.2. Second by the fact that by inserting the phase node the data points become ordered by time, which gives the edges a defined direction.

To analyze the found dependencies, the time courses of the expression levels of *YPL256C* and *YIL066C* were compared. As Figure 3.12 shows both curves are strongly correlated, but they do not reveal a causal relationship among those ORFs. There is no phase-shifting between the two curves, which may an indeed of causal dependency. Also in the Bayesian network structure without phase node the direction of the edge between *YPL256C* and *YIL066C* can not be obtained. This edge becomes only directed by an effect of third order, namely by the phase node and the resulting directed edge out of them from *YER001W* to *YPL256C*. This leads to the observation, that effects of higher order, which are not evident from a correlation analysis, can be revealed by the way of Bayesian statistics. But for better understanding of such effects, they should be examined more accurately in the future.

Another point, which arises from the nontrivial, is the fact that an edge goes from the phase node to *YER001W* instead to *YPL256C*, which oscillates more and from the phase node to *YGL021W* instead of the highly oscillated ORF *YMR001C*. Since both ORFs, which are connected with the phase node, are also highly oscillating as Figure 3.13 shows, one reason could be, that the learning procedure often stucked in a good but not in the best optimum.

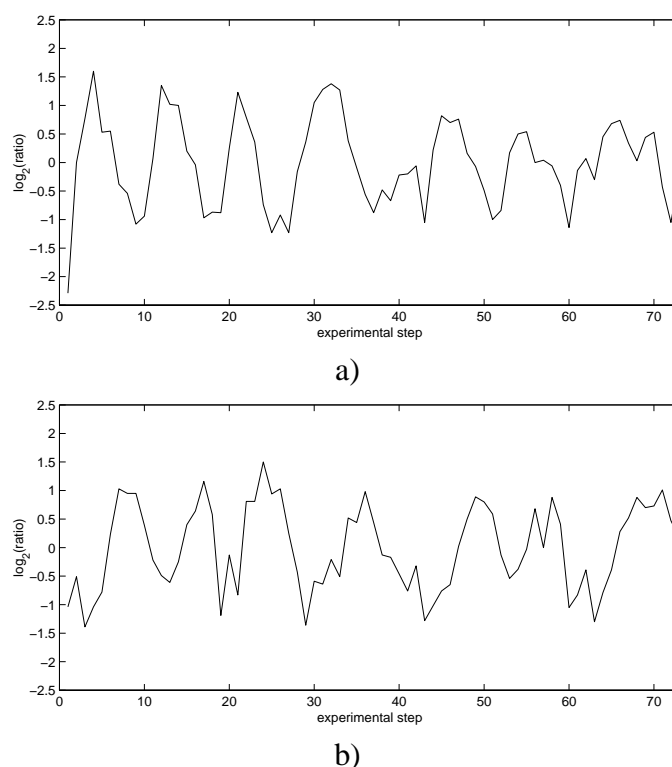


Figure 3.13: The time courses of the expression levels of ORF a) *YER001W* and b) *YGL021W*. Both are highly cell cycle regulated.

The *YOR263C*-subnetwork

This subnetwork, which was reconstructed from the results from Friedman et al. [16] consists of 8 ORFs and an additional phase node. In the clustering experiments of Spellman et al. most of them appear in the same cluster, namely in the *SIC1* cluster. Figure 3.14 shows the resulting fPDAG of 20 learned structures, which contains only undirected edges but some of them with a very high confidence level. The most conspicuous feature is the undirected edge between *YOR263C* and *YOR264W*. Both ORFs are located next to each other on the DNA strands of chromosome *XV* and both have unknown biological and molecular functions. Another high confidence feature is the undirected edge between *YNR067C* and *YGL028C*. Whereas the function of *YNR067C* is unknown *YGL028C* is known to be a soluble cell wall protein. It is also related to *YER124C*, the function of which is unknown, and to *YLR286C*, an endochitinase involved in cell wall biogenesis. This two ORFs are again connected by an undirected edge of high confidence.

Since *YER124C* is connected to two ORFs, both functionally related to cell wall

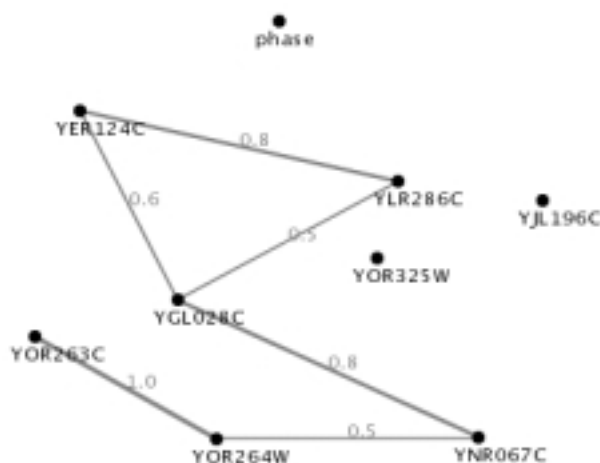


Figure 3.14: The resulting feature PDAG learned from the *YOR263C*-subnetwork; it consists of 8 ORFs and the phase node. The most interesting feature is the undirected edge between *YOR263C* and *YOR264W* with a confidence of 1.

biogenesis, one could assume that it is also involved in cell wall biogenesis. By this the Bayesian network trained here has provided a testable prediction of an unknown gene function so far. The two ORFs which are independent from the other nodes, namely *YJL196C* and *YOR325W* do not belong to the same cluster as the one, and therefore it seems reasonable that they are independent.

As already noted above, the learned subnetwork consists only of undirected edges, although a phase node was included. This may change by analyzing the complete dataset of 800 ORFs, since then other edges might have a defined direction and thereby in turn define the direction of the edges considered here.

The histone-subnetwork

This network consists of 9 ORFs which are known to express *histones*, and of a phase node. Histones are a specialized form of basic protein, present in all eukaryotic nuclei. They are associated with DNA in a structure called *chromatin*. Besides structural functions histones play a major role in the regulation of gene transcription, namely by inhibiting the interaction between the promoter DNA and general transcription factors. Thus they belong to the group of repressor proteins.

Figure 3.15 shows the resulting fPDAG of 20 learned structures, which is very sparse containing only 4, but high confident, edges. The structure displays a strong

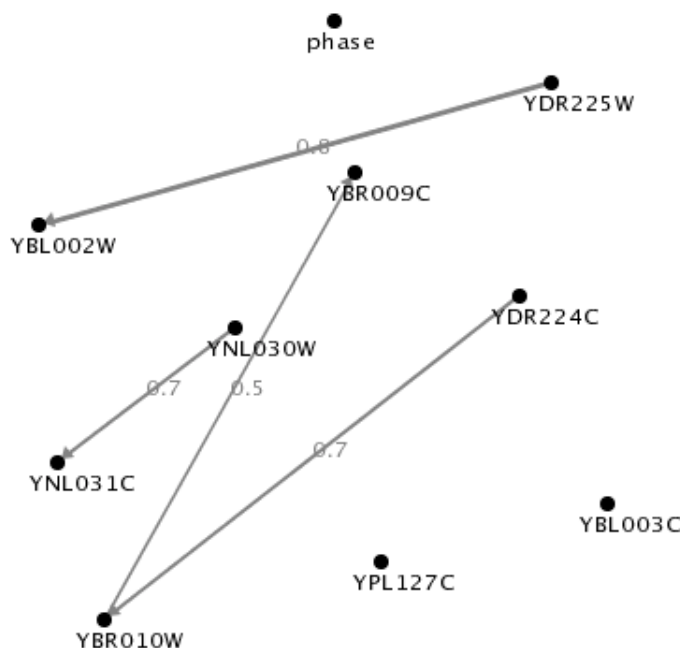


Figure 3.15: The resulting feature PDAG learned from the histone-cluster; it consists of 9 ORFs, which code for histone proteins, and the phase node.

dependency between *YDR225W* and *YBL002W*, which confirms the clustering result of Spellman et al., but furthermore it shows the causal relationship between those ORFs namely that *YDR225W* influences *YBL002W*.

The same holds for ORF *YDR224C* and *YBR010W*, where *YDR224C* affects *YBR010W* and for *YNL030W* and *YNL031C* where *YNL030W* influences *YNL031C*. The interesting thing about the last relation is the fact that the two ORFs, *YNL031C* and *YNL030W*, are located close to each other on the DNA strands chromosome *XIV* as shown in Figure 3.16. ORF *YNL030W* and *YNL031C* also referred as to *HHF2* and *HHT2* are located on opposite DNA-strands one after the other, and therefore they are likely to be expressed together. Also *YBR009C* and its parent node *YBR010W*, are located next to each other on the opposite strands of chromosome *II*.

This example demonstrates the advantages of Bayesian networks over other statistical techniques, such as clustering methods, already used in this context. Besides revealing correlation between genes, they also detect causal relationships between them, which is important for understanding the function of genes and

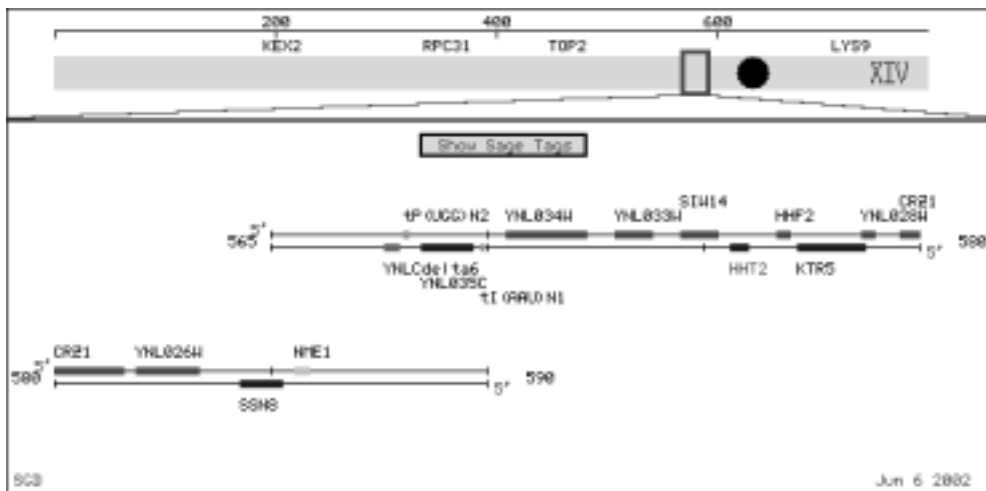


Figure 3.16: The chromosomal feature map taken from the SGD homepage [41]. ORF *HHT2* and *HHF2* are located on opposite DNA strands but next to each other.

their products. Moreover, Bayesian networks can detect suitable relationships and finer structures within a single cluster or subnetwork.

Finally it should be annotated, that analyzing only small subnetworks, as already done, can not reveal all relationships among genes, since it had not been proved that they are modular subnetworks in such a manner that they are independent from all other genes.

Chapter 4

Discussion

In this thesis the approach of learning Bayesian networks was applied on microarray datasets with the goal to reveal regulatory mechanisms among genes by modeling causal relationships. The implemented framework was applied to several small subnetworks, and the experiments showed that on the one hand the results correlate with those of clustering methods. On the other hand experiments showed that applying Bayesian networks can reveal new information about the function of genes, which are not available from clustering analyses. This is the case because a Bayesian network can describe arbitrary combinatorial control of gene expression and thus is not limited to pair-wise interactions between genes as in typical clustering methods, moreover because the learned networks have probabilistic semantics, which better fits the stochastic nature of both the biological processes and the noisy experiments and finally because it also can reveal causal patterns from gene expression data. The resulting networks, which are visualized graphically, are easy interpretable in a biological manner, and could be used by other scientists for further explorations.

The major problem of this framework is the sparseness of the data from microarray experiments. Thus a part of the thesis deals with the effect of data sparseness on the resulting learned networks. Results from benchmark datasets showed that with such small sample sizes as in at the time available microarray datasets the statistical content, which is used to model Bayesian networks, is so sparse, that only strong statistical relationships can be extracted. But as shown in section 3.1.2, the dependencies that are detected are relatively reliable even if the dataset is small. Since learning Bayesian networks is a purely statistical model, apart from prior knowledge about genetic pathways, it depends on the quality of the data. But nevertheless, also with such datasets it was possible to extract features, which are biologically reasonable, and since the price of microarray chips will decrease in the next time more samples are expected to be available which in term will allow to model more complex genetic networks than now in the near future.

Such models could then support the process of *drug discovery*, by detecting genes, which play an integral role in biochemical processes. In Bayesian networks such *dominant genes* can be easily detected since on the one hand they have no parents and on the other hand they have many children. Due to their dominant role in complex biochemical systems their malfunction may cause many diseases as for example tumor-specific abnormalities, which makes them important for pharmacology as *target genes* for new drugs.

Finally it should be annotated that this framework is not limited on modelling only relationships among genes, but it can also be used for analyzing protein-protein interactions by learning proteomic networks from protein datasets.

4.1 Future work

Since one of the major advantages of Bayesian network is the ability to combine prior knowledge with information extracted from data, integrating prior knowledge into the framework may improve the results. The Kyoto Encyclopedia of Genes and Genomes (KEGG) [28], for example, provides a big collection of graphical pathway maps of different organisms. Another interesting project is the gold database [18], which integrates disparate information on the function and properties of genes and their protein products, that are related to lipid-associated disorders. Including this and other biological knowledge in the framework would be a promising extension for modelling genetic networks.

A critical part of this framework is that a multinomial model was used. Since this model suffer from the information loss caused by discretization, applying a linear gaussian model should be discussed in the future.

Another critical part of this framework is the heuristic search strategy. Improving the theory and algorithms for heuristic search would be another important part of future work. In addition, for a significance analysis of the learned network structure it should be applied a *bootstrap method*, as in Friedman et al. [16] to get a more robust model.

Since common microarray datasets monitor thousands of genes the resulting space of possible networks structures is too large for an efficient heuristic search. Thus, to facilitate efficient learning, the search procedure has to focus on relevant regions of the search space by restricting the search space in a statistical manner, as done by Friedman et al. 1999 [17], or in a biological manner including prior knowledge.

Another drawback of this framework is that no dynamics can be modeled, and since most of the experiments produce temporal expression data, the existing framework should be extended for learning *Dynamic Bayesian networks*, as proposed by Murphy et al. [36].

Finally, another important question in the modeling of genetic networks, is to what extent the network is modular. If it would possible to get well specified subnetworks, e.g from clustering results, the learning procedure could be applied to such small subnetworks, as already done in section 3.2.3 but without information losses, and it would require much less data points than learning the complete genomic network.

4.2 Conclusion

In this thesis the basics of gene regulatory mechanisms and its importance for biomedical and pharmaceutic research was shown. The challenge for computational genomics is to discover ways to extract useful biological information from such data. The framework presented in this work found high regulating genes, which are already known, and it predicted the function of new genes unknown so far. So it may play an important role in revealing new biochemical pathway features as a part of the drug discovery process by finding new *target genes*.

4.3 Acknowledgments

I would like to thank my supervisor, Dr. Martin Stetter, for his guidance in my work, always giving me new ideas for this Master thesis.

I am also grateful to Prof. Schürmann, head of the Department for Neural Computation at Siemens Corporate Technology, for granting me the opportunity of writing my Master Thesis in a real research environment.

I also would like to thank Dr. Volker Tresp and his group, working on graphical models, machine learning and data mining, especially Dr. Michael Haft and Dr. Reimar Hofmann, for introducing me in the areas of graphical models and machine learning. I would also thank my other colleagues, Dr. mult. Gustavo Deco, Dr. Silvia Corchs and Norbert Galm, with whom I shared a good time and all other members of of the research group for the good and friendly atmosphere.

I am also grateful to Dr. Zlatko Trajanoski and the Bioinformatics Group at the Graz University of Technology, for their support and the opportunity to run my simulations on their Linux Cluster.

Chapter 5

Benchmark-Datasets

5.1 The Asia-network

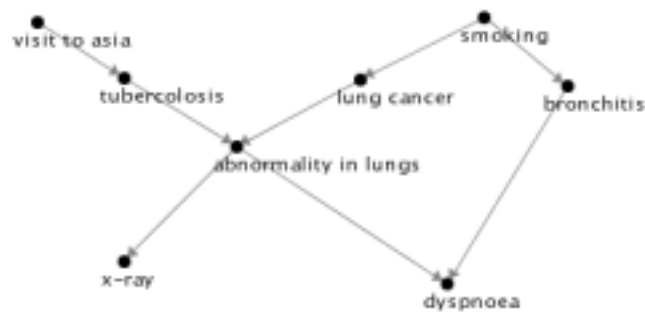


Figure 5.1: The Asia network encodes a fictitious medical system

This very small Bayesian network, proposed by Spiegelhalter et al. [46], encodes the probability distributions of a fictitious medical example, also called *chest clinic*. It consists of 8 discrete variables with binary values (true, false) connected by 8 edges. The variables represent diseases, e.g. bronchitis, or other conditions, e.g. if patient has been to Asia recently. The set of conditional probability distributions and the structure, used for generating the datasets are described on the *Netica* homepage [37].

5.2 The Alarm-network

This Bayesian network was constructed from expert knowledge as a medical diagnostic alarm message system for patient monitoring and has become the most popular benchmark-network for assessing structural learning algorithms. The domain has 37 discrete variables taking between 2 and 4 values, connected by 46 directed edges, which can be interpreted in a causal manner. The set of conditional probability distributions and the structure, used for generating the datasets are described on the *Netica* homepage [37].

Bibliography

- [1] T. Akutsu, S. Miyano and S. Kuhara (1999). Identification of genetic networks from a small number of gene expression patterns under the boolean network model. Pacific Symposium on Biocomputing'99 (PSB'99).
- [2] P. Baldi, S. Brunak (1998). Bioinformatics, the Machine Learning Approach. MIT Press.
- [3] I. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, in Proc. of the Second European Conf. on Artificial Intelligence in Medicine (London, Aug.), 38, 247-256. Also Tech. Report KSL-88-84, Knowledge Systems Laboratory, Medical Computer Science, Stanford Univ., CA.
- [4] M. Carey and S. Smale (2000). Transcriptional Regulation in Eukaryotes: Concepts Strategies and Techniques Regulation in Eukaryotes: Concepts Strategies and Techniques. Cold Spring Harbor Press: Cold Spring Harbor, NY.
- [5] J.E. Celis, M. Kruhøffer, I. Gromova, C. Frederiksen, M. Østergaard, T. Thykjaer, P. Gromov, J. Yu, H. Palsdottir, N. Magnusson, TF. Ørntoft (2000). Gene expression profiling: monitoring transcription and translation product using DNA microarrays and proteomics.
- [6] T. Chen, H.L. He, G.M. Church (1999). Modeling Gene Expression with Differential Equations. Proc. of Pacific Symposium on Biocomputing pp.29-40.
- [7] D.M. Chickering (1996). Learning Bayesian Networks from Data, UCLA Cognitive Systems Laboratory, Technical Report (R-245), June 1996. Ph.D. Thesis.

- [8] D.M. Chickering, D. Geiger, and D.E. Heckerman (1994). Learning Bayesian Networks is NP-Hard, Microsoft Research Technical Report MSR-TR-94-17.
- [9] Computational Modeling of Genetic and Biochemical Networks (Computational Molecular Biology), James M. Bower and Hamid Bolouri (Editors), 2001, MIT Press.
- [10] F.G. Cozman (2000). Generalizing variable elimination in bayesian networks. In Workshop on Probabilistic reasoning in Bayesian networks at SBIA/Iberamia 2000, pages 21–26.
- [11] M.H. DeGroot (1970). Optimal Statistical Decisions. McGraw-Hill, Inc..
- [12] A.P. Dempster, N.M. Laird, and D.B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, B, 39:1-38.
- [13] P. D’haeseleer, S. Liang and R. Somogyi (1999). Tutorial: Gene Expression Data Analysis and Modeling. Pacific Symposium on Biocomputing ’99 (PSB’99).
- [14] M.B. Eisen, P.T. Spellman, P.O. Brown and D. Botstein (1998). Cluster Analysis and Display of Genome-Wide Expression Patterns. Proc Natl Acad Sci U S A 95, 14863-8.
- [15] N. Friedman (1997). Learning Bayesian networks in the presence of missing values and hidden variables.
- [16] N. Friedman, M. Linial, I. Nachman, and D. Pe’er (2000). Using Bayesian Network to Analyze Expression Data. Journal of Computational Biology, Vol. 7, pp. 601-620.
- [17] N. Friedman, I. Nachman, and D. Pe’er (1999). Learning Bayesian network structure from massive datasets: The ”sparse candidate” algorithm. In Proc. UAI.
- [18] Gold Databse website.
<http://gold.tugraz.at>
- [19] D. Heckerman, D. Geiger, and D. Chickering (1995). Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, 20(3):197-243, Kluwer.

- [20] D. Heckerman, D. Geiger, and D. Chickering (1994). Learning Bayesian networks: The Combination of Knowledge and Statistical Data. Technical Report MSR-TR-94-09, Microsoft Research.
- [21] R. Hofmann (2000). Lernen der Struktur nichtlinearer Abhängigkeiten mit graphischen Modellen. Doctoral thesis.
- [22] R. Hofmann, V. Tresp (1996). Discovering Structure in Continuous Variables Using Bayesian Networks. In Touretzky, D. S., Mozer, M. C., Hasselmo, M. E. (Hrsg.), *Advances in Neural Information Processing Systems 8*, MIT Press.
- [23] R. Hofmann, V. Tresp (1998). Nonlinear Markov Networks for Continuous Variables. In Jordan, M. I., Kearns, M. S., Solla, S. A., (Hrsg.), *Advances in Neural Information Processing Systems 10*, MIT Press.
- [24] Human Genome Project website.
<http://www.ncbi.nlm.nih.gov/genome/guide/human/>
- [25] T. Ideker, V. Thorsson, J.A. Ranish, R. Christmas, J. Buhler, J.K. Eng, R. Bumgarner, D. Goodlett, R. Aebersold, L. Hood, (2001). Integrated genomic and proteomic analysis of a systematically perturbed metabolic network. *Science* 292: 929-934.
- [26] K.M. Kerr, M. Martin, G.A. Churchill (2000). Analysis of Variance for Gene Expression Microarray Data. *J Comput Biol.* 2000;7(6):819-37.
- [27] S. Kirkpatrick, C.D.Jr. Gerlatt and M.P. Vecchi (1983). Optimization by Simulated Annealing, *Science* 220, 671-680.
- [28] Kyoto Encyclopedia of Genes and Genomes website.
<http://www.genome.ad.jp/kegg/>
- [29] P.J.M. Van Laarhoven, E.H.L. Aarts, Dordrecht (1987) *Simulated annealing: theory and applications.*
- [30] S.L. Lauritzen (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201.
- [31] R.J.A. Little and D.B. Rubin (1987). *Statistical Analysis with Missing Data.* New York: John Wiley.

- [32] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller (1958). Equations of State Calculations by Fast Computing Machines, *J. Chem. Phys.* 21, 1087- 1092.
- [33] Microarray dataset, from Standford Universisty.
<http://genome-www.stanford.edu/cellcycle/>
- [34] Molecular Cell Biology, Fourth Edition Harvey Lodish (Massachusetts Institute of Technology), Arnold Berk (U. of California, Los Angeles) Lawrence Zipursky (UCLA), Paul Matsudaira (MIT), David Baltimore (California Institute of Technology), James Darnell (Rockefeller U.).
- [35] K. Murphy (2001). An introduction to graphical models. Technical report, Intel Research Technical Report.
- [36] K. Murphy, S. Mian (2000). Modelling Gene Expression Data using Dynamic Bayesian Networks. Technical Report, University of California, Berkeley.
- [37] Network library of Norsys Software Corp..
<http://www.norsys.com/netlib/>
- [38] J. Pearl. (1997). Graphical models for probabilistic and causal reasoning. *The Computer Science and Engineering Handbook*, pages 697–714.
- [39] M. Pincus (1970). A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems, *Oper. Res.* 18, 1225-1228, 1970.
- [40] Ramser, et al. (2001). A physical map of the human genome *Nature* 409, 934-941. The Human Genome Project
- [41] Saccharomyces Genome Database website.
<http://genome-www.stanford.edu/Saccharomyces/>
- [42] E. Sakamoto and H. Iba (2001). Inferring a System of Differential Equations for a Gene Regulatory Network by using Genetic Programming. *Congress on Evolutionary Computation*, pp.720-726, IEEE Press.
- [43] M. Schena, D. Shalon, R. Davis and P.O. Brown (1995). Quantitative monitoring of gene expression patterns with a cDNA microarray. *Science* 270:467-470.
- [44] G. Schwarz (1978). Estimating the dimension of a model. *Annals of Statistics*, 7(2):461-464.

- [45] Spellman et al. (1998). Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization. *Molecular Biology of the Cell* 9, 3273-3297.
- [46] Spiegelhalter, David J. (1986). Probabilistic reasoning in predictive expert systems in *Uncertainty in Artificial Intelligence*, L.N. Kanal and J.F. Lemmer (Eds.), North-Holland, Amsterdam, pp. 47-67.
- [47] R. Somogyi, S. Fuhrman, M. Askenazi, A. Wuensche (1997). The Gene Expression Matrix: Towards the Extraction of Genetic Network Architectures. *Nonlinear Analysis, Proc. of Second World Cong. of Nonlinear Analysts (WCNA96)* 30(3):1815-1824.
- [48] H. Steck, V. Tresp (1999). Bayesian Belief Networks for Data Mining. In *Proc. 2. Workshop Data Mining und Data Warehousing als Grundlage moderner entscheidungsunterstützender Systeme, (DMDW99), LWA99 Sammelband*, Univ. Magdeburg.
- [49] H. Steck (2001). Constraint-Based Structural Learning in Bayesian Networks using Finite Data Sets. Doctoral thesis.
- [50] V. Tresp, R. Hofmann (2001). Nonlinear Time-Series Prediction with Missing and Noisy Data. In Jordan, M. I. and Sejnowski, T. J. (Hrsg.), *graphical models: foundations of neural computation*, MIT Press.
- [51] T. Verma and J. Pearl (1990). Equivalence and synthesis of causal models. In *Proceedings of Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227.
- [52] Jason Tsong-Li Wang, Bruce A. Shapiro, Dennis Shasha: *Pattern Discovery in Biomolecular Data: Tools, Techniques and Applications*. Oxford University Press 1999