Finding Optimal Models for Gene Networks

(遺伝子ネットワークにおける最適なモデルの探索)

Sascha Ott

A Dissertation
博士論文

February 27th 2004

Thesis Supervisor: Professor Satoru Miyano (宮野悟)
Human Genome Center, Institute of Medical
Science, The University of Tokyo

ABSTRACT

In recent years, the entire hereditary information of complex organisms like humans has become available. This information is organised in genes which are coded in the DNA and expressed by the processes of *transcription* and *translation*. By transcription, genes are replicated to RNA molecules. The major part of the RNA molecules is subsequently used as templates to form a number of protein molecules by translation. The RNA molecules and the proteins fulfill diverse functions within the cell, and the cooperation of these molecules is the fundament of life. Controlling transcription and translation of all genes allows cells to grow, to divide, and to adapt to changing environmental conditions. Since the control mechanisms are also fulfilled by RNA molecules and proteins, the expression and regulation of genes forms a cycle of genes regulating genes mutually. We call the entirety of these interactions the *gene network*.

With the invention of new technologies such as DNA microarrays, it has become possible to measure the expression levels of most genes of a cell simultaneously. This has created a need for methods to extract information about gene networks from such data. Gaining such information would be a fruitful contribution to molecular biology, medicine, pharmacy, and other areas. Hence, the estimation of gene networks from gene expression measurements has become one focus of bioinformatics research.

However, the problem of estimating gene networks is NP-hard and the search space is of super-exponential size. Applying a brute force approach in order to overcome these problems would be prohibitive even for gene networks of 9 genes. If instead heuristic search is applied, the accuracy of gene network estimations becomes uncertain. Therefore, optimal gene network models could so far not be obtained.

Though a considerable amount of research work has been spent on the estimation of gene networks, there has been little work on the problem to evaluate the quality of these estimations in a principled way. Problems of such evaluations are, for example, the fragmentary knowledge about gene networks and the uncertainty about what part of a gene network is working under given experimental conditions. As a result, a firm proof of the significance of gene network estimations had not been done before.

We have analysed the structure of the search space and developed an algorithm that finds the optimal solution within the super-exponential search space in exponential time. This approach is feasible for gene networks of 20 or more genes, depending on the modelling approach employed. Adding a biologically justified assumption, the optimal network can be found for gene networks of up to about 35 genes. Furthermore, since in real data gene expression patterns do not show significant differences in quite a few cases, many genes can be and should

be grouped to what we denote as *network elements*. Building gene networks of these essential elements, optimal gene networks can be found for sets of about 100 genes in practical applications.

Therefore, this algorithm is a valuable tool for extracting information about gene networks from gene expression measurements. Overcoming the uncertainties of heuristics also opens up the possibility to compare statistical modelling approaches with respect to their power to infer biologically accurate gene networks. Similarly, data sets obtained with different experimental techniques can be assessed for their information content. Since the algorithm is based on a general definition of the gene network estimation problem, it can be applied to all existing gene network modelling approaches and it does not depend on a certain kind of gene expression data.

However, while optimal gene network models are the models with the best score with respect to some score function, there may still be significantly different models that have approximately the same score with respect to given data, especially since gene expression measurements will in general only provide partial information about the entire gene network. Furthermore, even for a gene network of 10 genes, there are about $4.17 \cdot 10^{18}$ possible network models. Therefore, even optimal gene network models will in general not match the target network.

In this work, we tackle this problem in two ways. First, we extent the theoretical basis and the algorithm mentioned above in order to allow for the enumeration of optimal and suboptimal networks in the order of their score. This algorithm is applicable for gene networks of about the size discussed above and can enumerate the best m network models, where m may be as large as $20,000$. Second, we apply several approaches to extract the most reliable part of gene network models from a given list. We denote these partial networks as *gene network motifs*. Our results show that gene network motifs identified by our algorithms are in significantly better agreement with the knowledge than optimal network models. We also apply this approach to the available data of *Bacillus subtilis* and *Escherichia coli* and do the first comparison of gene networks of related species based on estimations.

Finally, we generalise above techniques to gene networks of arbitrary size by making use of the prevalent assumption of compartmentalisation of gene networks in dense components which are sparsely connected to each other. This is done by identifying a biologically relevant subspace of the search space, and apply the techniques mentioned above to the subspace. Subspace selection can be done by making use of biological knowledge or, if no previous knowledge is available, by applying clustering methods or heuristic algorithms. The running time needed to find optimal solutions in a subspace is $O(n)$, where n is the number of genes, therefore allowing for the estimation of arbitrarily large gene networks.

Comparison of estimated large gene networks for to biological knowledge yields a significant agreement.

In some of the evaluations of this thesis, we rigorously compare estimated gene network models to the available knowledge about true gene networks. We use several approaches to do the comparison and each of these approaches consistently yields highly significant agreement. Therefore, as an overall result of this thesis, the significance of gene network estimations is proved.

We also include some results about the estimation of gene networks in the special case of using differential equations for the analysis of time-course data. Based on synthetic expression data, we evaluate the influence of the magnitude of measurement error and the number of measurements on the estimation accuracy. Furthermore, we show the results of a comparison of estimations based on differential equations to estimations based on the BNRC score for Bayesian networks. Finally, in order to classify the gene network estimation problem in the special case of modelling with differential equations, we show the NP-hardness based on a definition of the problem in this case.

論文要旨

近年、人間などの高等生物の遺伝情報を完全に解読することが可能になってきている。遺伝情報の単位である遺伝子がDNAに暗号化されており、**転写**と**翻訳**によって発現する。転写によって遺伝子がRNAへと複製されてから、RNAを鋳型に蛋白質が翻訳によって作られる。RNAと蛋白質の分子が細胞の中の多様な機能を実現しており、生命の現象の基本となっている。全遺伝子の転写と翻訳を調整することによって、細胞の成長、細胞の分裂、環境条件の変化に対応することなどが可能になる。その調整を果たしているのもRNAと蛋白質であるので、遺伝子の発現と遺伝子の調整がサイクルを形成し、それぞれの遺伝子の発現レベルは相互に調整される。その様な発現レベルから見た依存関係の全貌をここでは**遺伝子ネットワーク**とよぶ。

DNAマイクロアレイ等の新しい技術の開発により、細胞の殆どの遺伝子の発現レベルを同時に測ることが可能になってきている。その様な一連の実験で得るデータから遺伝子ネットワークについての情報を抽出する方法は、ポストゲノム時代の最優先のニーズの一つである。遺伝子ネットワークの情報を入手できれば、細胞システムの理解が深まり、分子生物学、医学、製薬等への貢献をもたらすと期待される。そのため、遺伝子ネットワークの発現データからの推定が生物情報科学の主なテーマの一つとなっている。

しかし、遺伝子ネットワークの探索問題はNP困難な問題であり、探索空間の大きさは超指数関数的である。ブルートフォース的な方法の適用では、９個の遺伝子のネットワークの場合でも実現できない規模の計算量となる。これらの問題を避けて、ヒューリスティクなアルゴリズムを適用すると探索の結果の良さが不明になる。そのため、今までは確実に最適な遺伝子ネットワークのモデルを得ることは不可能だった。

遺伝子ネットワークが注目を集める問題であるのに、今までの研究ではネットワークの推定について、本質的な評価が行われなかった。推定の評価を行う際、問題になるのは、実際の遺伝子ネットワークの知識が断片的であることやネットワークのどの部分がどういう実験の条件で働くかという不確かさ等である。それゆえ、発現データから推定される遺伝子ネットワークのモデルが有意であることの証明は今までになかった。

本研究ではネットワークの探索空間の解析を行い、指数関数的な時間で超指数関数的な探索空間の中の最適なネットワークを見付け出すことのできるアルゴリズムを開発した。これにより、遺伝子数が２０個の場合でもこのアルゴリズムを現実的な時間で利用することが可能となった。さらに生物学的に妥当な制約条件のもとでは、遺伝子数が３５個前後の時でも最適なモデルの探索が行える。また、実際の発現データでは有意に違わない発現のパターンを見せる遺伝子が多いということを考慮して、パターンが等しいといえる遺伝子をグループにまとめることができる。そういった本質的な部分を**ネットワーク要素**とよぶ。遺伝子ネットワークをネットワーク要素から構成して１００個前後の遺伝子を扱って、最適なモデルの探索が現実的に可能であることを明らかにする。

したがって、遺伝子数が限られた場合においては、発現データから遺伝子ネットワークについて情報を得ることができるというメリットをこのアルゴリズムはもっている。このアルゴリズムが発現データの種類にもこれまで提案されている統計学的な遺伝子ネットワークのモデリングの方法にも依らないので、一般的に利用することが可能である。ヒューリスティクなアルゴリズムの不確実な結果を避け、こうした統計学的なモデリングの方法のそれぞれに対して最適なモデルの探索を行うことにより、モデリングの方法の適切さを評価することができる。同様に、異なる実験の方法を適用して得たデータのそれぞれに対して最適なモデルの探索を行えば、遺伝子ネットワークを発見するための実験方法の良さも評価でき、データの良い組み合わせ方も解析できる。

ところが、最適なネットワークとは、モデリングに対応したスコア関数に関して最適なネットワークに過ぎない。ほぼ等しいスコアで構造の異なるネットワークが数多く存在しうる。その理由に依り、データにはそもそもネットワーク全体の情報は無くて、全ネットワークの部分的な情報しかないという可能性を十分に考慮するべきである。遺伝子数またはネットワーク要素数が１０個のときでもネットワークの候補の数が $4.17 \cdot 10^{18}$ の規模であるので、最適ネットワークといっても、生物学的に真の遺伝子ネットワークとは違うことが多々ある。

本研究ではその問題に取り組んで、二段階の方法を開発した。一つ目に、上述のアルゴリズムの理論を更に展開させて、最適なモデルから始めてネットワークをスコアの順で数え上げることができるようにした。数え上げの行えるネットワーク要素数は上に述べた規模とほぼ同じであるし、m が２万のときでも一番スコアの良い m 個のネットワークを現実的に数え上げられる。二つ目に、スコアの良いネットワークを比較して、共通な部分を取り出せる幾つかの方法を適用する。この様な共通部分を**ネットワークモチーフ**とよぶ。研究の成果として、実際のデータを使って推定したネットワークから抽出したネットワークモチーフの方が最適なネットワークより生物学的な知識と有意に一致していることがわかった。数え上げとネットワークモチーフ抽出からなる方法により、枯草菌と大腸菌を用いて、本研究で初めてネットワーク推定に基づいた遺伝子ネットワークの比較を行った。その解析の結果も紹介する。

さらに、上述の方法を任意に大きな遺伝子ネットワークの場合にも適用できるアプローチを提案する。ここで、広く認知されている遺伝子ネットワークの一般的な構造、すなわち密度の高い構成成分が存在し、その構成成分同士はまばらにつながっている構造を前提する。この前提のもとで生物学的に意味のある探索空間の部分を選択してから、部分探索空間に最適なモデルの探索アルゴリズム、数え上げのアルゴリズム、モチーフ探索アルゴリズムが適用できることを示す。その部分探索空間を選択するために、生物学的な既知の知識を適用できる。既知の知識が十分でない場合、クラスタリングまたはヒューリスティクなアルゴリズムを利用できる。ネットワーク要素数が n のとき、部分探索空間を探索する時間は $O(n)$ になるので、このアプローチにより要素数の制約を解決できる。この方法を利用して生物学的な知識と有意に一致する推定ができることを示す。

本研究で幾つかの手法を使って、推定したネットワークのモデルを生物学的な知識と厳密に比較した。いずれの手法でも推定が有意に知識と一致することが明らかになったので、研究の成果として遺伝子ネットワーク推定の意義を示すことができた。

また、時系列データからの推定に微分方程式を使うという特殊ケースについて得た結果も含める。人工的な発現データを作って、データのエラーの大きさと測定の数とは、推定の良さにどの様に影響を与えるかについて調べた。さらに、微分方程式のスコア関数と BNRC というスコア関数をそれぞれ同じデータに対して利用した結果を議論する。また、微分方程式を使った遺伝子ネットワーク推定問題の特殊ケースの複雑さを決めるために、その問題がある定義のもとで NP-困難であることを証明する。

Contents

Chapter 1

Introduction and Overview

On the quest for the understanding of life, mankind achieved a major advance in 1665, when the existence of cells was discovered [Hooke1665]. Later, the organisation of cells in organelles, the variety of different cell types, and the coexistence and communication of cells in multicellular organisms were intensively studied, and understanding of life was deepened to smaller and smaller scales of cellular constituents. Starting with the discovery of the molecular structure of DNA in 1953, the basic principles of storage, replication, and modification of hereditary information were revealed [Watson53]. Reading this information became possible in 1975, when DNA sequencing techniques were developed simultaneously by two independent groups [Maxam77, Sanger75].

Today, the entire hereditary information of complex organisms like humans is available, and the basic organisation of this information in genes as well as the way genes are expressed in living cells are understood. Genes are coded in the DNA and replicated to RNA by the process of *transcription*. The major part of the RNA molecules is subsequently copied to a number of protein molecules by the process of *translation*. Some RNA molecules and most protein molecules fulfill functions within the cell, and the cooperation of these molecules is the fundament of life. Controlling transcription and translation of all genes allows cells to grow, to divide, and to adapt to changing environmental conditions in order to sustain an equilibrium. Since the control mechanisms are also fulfilled by RNA molecules and proteins, the expression and regulation of genes forms a cycle of genes regulating genes mutually. We call the entirety of these interactions the *gene network*.

With the recent invention of new technologies such as DNA microarrays and protein chips, it has become possible to measure the expression levels of most genes of a cell simultaneously. This has created a need for methods to extract information about gene networks from such data. If at least partial information

about gene networks can be gained, this is supposed to be a fruitful contribution to molecular biology, medicine, pharmacy, and other areas. It may also lead to a better understanding of cellular processes, the responses of cells to their environment, and the construction principles of gene networks [Hartwell99, Shen-Orr02]. Hence, the estimation of gene networks has become one focus of bioinformatics research in recent years [Chen99, D'haeseleer99, Friedman00, Hartemink01, deHoon03a, Imoto02, Nariai04, V.A.Smith02, Someren02, Tamada03].

Measuring the Expression of Genes

The most frequently used technique among the currently available high throughput techniques for measuring the expression levels of genes is the technique of DNA microarrays. DNA microarrays are plates with DNA molecules attached to the surface. Since these DNA molecules are chosen as complementary to parts of the RNA molecules present in the cell, they can hybridise to specific target molecules. Different DNA molecules are placed on different spots of the microarray, each spot representing one gene. Since the number of molecules that are attached to one spot may vary substantially, DNA microarray experiments are conducted in a comparative fashion. Cells from two cell cultures are selected, RNA molecules are extracted from these, and a microarray is used to approximately determine the ratio of abundant RNA molecules of each gene in the two cell cultures. In most experiments altered cell cultures are compared to a reference cell culture. This allows to observe the changes of gene expression patterns under different conditions.

We note that gene expression levels can also be measured on the protein level. Though both kinds of data can be regarded as gene expression measurements and the algorithmic methods in this thesis can be applied to both, the resulting gene network estimation may be different because of the different regulatory mechanisms that are observable from these data (exemplified in [Wagner01]). For example, regulation on the level of translation will not be observable from RNA measurements.

Problems of the Estimation of Gene Networks

In order to estimate gene networks from expression measurements, gene networks have to be mathematically described, and a score function has to be defined to evaluate possible gene network models. Then, a search for the most likely network has to be conducted in the space of possible networks. However, this search problem is NP-hard (see Chapter 2) and the search space is of super-exponential size [Robinson73]. Applying a brute force approach in order to overcome these problems would be prohibitive even for gene networks of 9 genes. If instead heuristic search is applied, the accuracy of gene network estimations becomes uncertain. Therefore, optimal gene network models could so far not be obtained.

Though a considerable amount of research work has been spent on the estimation of gene networks, there is little work on the problem of the evaluation of the quality of these estimations in a principled way. To our knowledge, there has been only one publication that gives a well-defined p-value to prove the significance of estimations, though this is done for gene networks of restricted structure [deHoon03b].[1] Several problems are encountered when doing a principled evaluation of gene network estimations. Examples are the fragmentary knowledge about gene networks, and the uncertainty about what part of a gene network is working under what kind of experimental conditions. Furthermore, if structural differences between the true gene network and the estimated network are found, it should be distinguished between substantially wrong differences and admissible differences like transitive edges. As a result of these problems, a firm proof of concept for gene network estimation had not been done before.

Contributions of this Thesis

Given a search space of super-exponential size, researchers have been applying heuristic approaches to estimate gene networks. However, the accuracy of heuristics is uncertain, which - in combination with the high measurement noise of microarrays - makes it very difficult to draw conclusions from networks estimated by heuristics. In order to overcome this problem, we have analysed the structure of the search space and developed an algorithm that finds the optimal solution within the super-exponential search space in exponential time. This approach is feasible for gene networks of 20 or more genes, depending on the modelling approach used. Adding a biologically justified assumption, the optimal network can be found for gene networks of up to about 35 genes. Furthermore, since in real data gene expression patterns do not show significant differences in quite a few cases, many genes can be and should be grouped to what we denote as *network elements*. Building gene networks of these essential elements, optimal gene networks can be found for sets of about 100 genes in practical applications.

Therefore, this method is a valuable tool for extracting information about gene networks from gene expression measurements. Overcoming the uncertainties of heuristics also opens up the possibility to compare statistical modelling approaches with respect to their power to infer biologically accurate gene networks. Since the algorithm is based on a general definition of the gene network estimation problem, it can be applied to all existing gene network modelling approaches and it does not depend on a certain kind of gene expression data. The algorithm and the first result of its application to real data have been published in [Ott04a]. The algorithm is introduced in Chapter 3 of this thesis, where we

[1] In the evaluation in [deHoon03b], the number of parents of a gene was restricted to 1 and the number of genes was restricted to 7, limiting the scope of such evaluations to gene networks of a very restricted structure.

also discuss the algorithm‘s merits and further application results.

However, while optimal gene network models are the models with the best score, there may still be significantly different models that have approximately the same score with respect to given data, especially since gene expression measurements will in general only provide partial information about the entire gene network. Furthermore, even for a gene network of 10 genes, there are about $4.17 \cdot 10^{18}$ possible network models [Robinson73]. Therefore, even optimal gene network models will in general not match the target network.

In this work, we tackle this problem in two ways. First, we extent the theoretical basis and the algorithm mentioned above in order to allow for the enumeration of optimal and suboptimal networks in the order of their score. This algorithm is applicable for gene networks of about the size discussed above and can enumerate the best m network models, where m may be as large as $20,000$. Second, we apply several approaches to extract the most reliable part of gene network models from a given list. We denote these partial networks as *gene network motifs*. Our results show that gene network motifs identified by our algorithms are in significantly better agreement with the knowledge than optimal network models. We also apply this approach to the available data of *Bacillus subtilis* and *Escherichia coli* and do the first comparison of gene networks of related species based on estimations. A part of the results on network model enumeration and gene network motifs forms the content of [Ott04b]. Chapter 4 covers the extension towards the enumeration of the best m network models. Results concerning gene network motifs are the content of Chapter 5.

Finally, we generalise above techniques to gene networks of arbitrary size by making use of the prevalent assumption of compartmentalisation of gene networks in dense components which are sparsely connected to each other [Hartwell99, Someren02]. This is done by identifying a biologically relevant subspace of the search space, and apply the techniques mentioned above to the subspace. Subspace selection can be done by making use of biological knowledge or, if no previous knowledge is available, by applying clustering methods or heuristic algorithms. The running time needed to find optimal solutions in the selected subspace is $O(n)$, where n is the number of genes, therefore allowing for the estimation of arbitrarily large gene networks. Comparison of estimated large gene networks to biological knowledge yields a significant agreement. A part of the results on large gene networks is summarised in [Ott03]. We introduce several algorithms following above scheme and analyse estimation results of these algorithms in Chapter 6.

In some of the evaluations of this thesis, we rigorously compare estimated gene network models to the available knowledge about true gene networks. We use several approaches to do the comparison and each of these approaches consistently

yields highly significant agreement. Therefore, as an overall result of this thesis, the significance of gene network estimations is proved.

We also include some results about the estimation of gene networks based on modelling with differential equations in the special case of analysing time-course data. We evaluate the influence of the magnitude of measurement error and the number of measurements on the estimation accuracy. These evaluations are conducted based on synthetic gene expression data. Furthermore, we show the results of a comparison of estimations based on differential equations to estimations based on the BNRC score (introduced in Chapter 2). Finally, in order to classify the gene network estimation problem in the special case of modelling with differential equations, we show the NP-hardness based on a definition of the problem in this case. Some of these results were published as a part of [deHoon03b]. In this thesis, we include our results on differential equation models in Chapter 2, where we also give an introduction to modelling and estimation of gene networks.

Chapter 2

Modelling and Estimation of Gene Networks

The natural phenomenon of gene networks has to be mathematically described in order to allow for an estimation. Several ways of modelling have been proposed and applied to real data. In this chapter, we introduce these approaches and compare them. In order to tackle the gene network estimation problem in a general way, we then give a definition of the problem that captures all existing modelling approaches.

After introducing some basic terms in Section 2.1, we introduce the existing approaches of modelling gene networks in Section 2.2. In the same section, we also show results of an evaluation of the effect of measurement errors and the number of measurements on the estimation accuracy in the case of using differential equations. Then we introduce frequently used score functions for Bayesian networks in Section 2.3, which is the modelling approach used in most estimations of this thesis. We include a comparison of the accuracy of one of these score functions to a score function based on modelling with differential equations. In Section 2.4, we give the definition of the gene network estimation problem and discuss previous work on this problem. We conclude this chapter in Section 2.5.

2.1 Basic Terms and Definitions

We start with the definition of directed and undirected graphs. Note that we use the character G to denote a set of genes in this work, while we use the character N to denote graphs.

Definition 1:
Let G be a finite set. Let $G^{(2)}$ denote the set of all subsets of G with two elements.

Let $E \subseteq G \times G$ and $U \subseteq G^{(2)}$. We call the pair (G, E) a *directed graph*, and the pair (G, U) an *undirected graph* on G. •

Our definition of graphs does not allow self-loops in undirected graphs. We will also use the term "network" alternatively to "graph".

Definition 2:
Let $N = (G, E)$ be a directed graph, $n \in \mathbb{N}$ [1], $w = ((g_1, h_1), \ldots, (g_n, h_n)) \in E^n$ be an n-tuple, and let $g, h \in G$. We call w a *directed path* from g to h if the following conditions hold.

1. $g = g_1$
2. $h_i = g_{i+1}$ for all $i = 1, \ldots, n-1$
3. $h = h_n$

If there is a set of bijections $f_i : \{g_i, h_i\} \to \{g_i, h_i\}$ $(i = 1, \ldots, n)$ such that $((f_1(g_1), f_1(h_1)), \ldots, (f_n(g_n), f_n(h_n)))$ is a directed path from g to h, we call w an *undirected path* from g to h. •

We will also need the notion of directed acyclic graphs.

Definition 3:
Let $N = (G, E)$ be a directed graph. We call N *acyclic* if for all $g \in G$ there is not directed path from g to g. •

Since in the context of gene networks, one frequently focuses on the parents of a given gene, we define the following annotation.

Definition 4:
Let $N = (G, E)$ be a directed graph and $h \in G$. We define

$$Pa^N(h) = \{g | (g, h) \in E\}$$ •

In a cyclic graph, the following notion is useful to identify cyclic components.

Definition 5:
Let $N = (G, E)$ be a directed graph and $C \subseteq G$. We call C a *strongly connected component* if the following conditions hold.

[1] We define $\mathbb{N}$ as $\{1, 2, \ldots\}$ in this thesis.

1. For all $g, h \in C$ with $g \neq h$, there is a directed path from g to h.
2. C is a maximal set with this property. •

We also need the notion of bipartite graphs.

Definition 6:
Let $N = (G, E)$ be a directed graph. Let $A, B \subseteq G$ with $A \cap B = \emptyset$ and $A \cup B = G$. We call N *bipartite* with respect to A and B if $(A \times A) \cap E = \emptyset = (B \times B) \cap E$. •

Definition 7:
Let $N = (G, E)$ be an undirected graph. Let $A, B \subseteq G$ with $A \cap B = \emptyset$ and $A \cup B = G$. We call N *bipartite* with respect to A and B if for all $e \in E$, $e \not\subseteq A$ and $e \not\subseteq B$. •

Definition 8:
Let $N = (G, E)$ be a directed graph. Let $f : G \to \{1, \ldots, |G|\}$ be a bijection. We call f a *topological sort* of N if $f(g) < f(h)$ holds for every $(g, h) \in E$. •

As a conclusion from Definition 8, we see that topological sorts can only be found for acyclic graphs.

Definition 9:
Let $n \in \mathbb{N}$ and $x, y \in \mathbb{R}^n$. Let $\bar{x}$ denote $\frac{1}{n} \cdot \sum_{i=1}^{n} x_i$, σ_x denote

$$\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^2},$$

and $\bar{y}$ and σ_y accordingly. The *Pearson correlation coefficient* of x and y is

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{\sigma_x} \cdot \frac{y_i - \bar{y}}{\sigma_y} \right)$$ •

We conclude this section with the definition of the gamma function.

Definition 10:
The *gamma function* $\Gamma : \mathbb{R} \to \mathbb{R}$ is defined as $\Gamma(x) = \int_0^\infty t^{x-1} \cdot e^{-t} dt$. •

2.2 Modelling Gene Networks

We introduce and discuss the existing[2] approaches to model gene networks. Gene networks have been mathematically described as Boolean networks [Akutsu99, Akutsu00a, Somogyi96], by using differential equations [Chen99, deHoon03a], and as Bayesian networks [Friedman98b, Friedman00, Imoto02, Pe'er01]. In all these cases, a gene network model is a mathematical object that can be described by a directed graph and a set of parameters. However, in this thesis we will make use of the term "gene network model" in two ways. Sometimes, we will use "gene network model" to denote the entire model, and sometimes to denote only the network (the graph) itself. The meaning will be clear from the context. We note that also Petri nets are used to describe gene networks (and also metabolic networks), but this usage is so far restricted to the simulation of known networks [Matsuno00].

Throughout the rest of this chapter, we assume we are given a set of genes $G = \{g_1, \ldots, g_n\}$. We start with introducing Boolean network models in Subsection 2.2.1. In Subsection 2.2.2, we continue with a discussion of using differential equations for describing gene networks, which also includes an evaluation of the influence of measurement error and the number of measurements on estimation results. We complete this section by introducing the basic concept of Bayesian networks in Subsection 2.2.3.

2.2.1 Boolean Networks

Boolean network models [Akutsu99, Liang98, Somogyi96, Szallasi98] only allow two states of gene expression levels. These two states correspond to an ON-state and an OFF-state. Therefore, in order to use Boolean network models, gene expression data has to be discretised down to two values using some empirical threshold. Even for genes that show a behaviour justifying the two-state-assumption, appropriate discretisation thresholds may be different for different genes. Therefore this assumption is strong, may be not biologically justified, and causes practical problems that may potentially bias the estimation results.

After discretisation, the gene expression data can be considered as a data set D of m Boolean vectors $v^{(1)}, \ldots, v^{(m)} \in \mathbb{B}^n$, where m denotes the number of experiments, n denotes the number of genes, and $\mathbb{B}$ equals $\{0,1\}$. A Boolean gene network model is a pair (N, F) of a directed graph $N = (G, E)$ and a tuple of Boolean functions $F = (f_1, \ldots, f_n)$ that complies with the following restriction. For every $i = 1, \ldots, n$, f_i has the functionality $\mathbb{B}^{q_i} \to \mathbb{B}$, where q_i abbreviates

[2] We note that there is one publication proposing the use of neural networks for gene network estimation [Mjolsness00]. We do not cover this approach here, since the proposed neural network only consists of two layers (the input layer and the output layer) and is, therefore, very similar to modelling by differential equations.

$|Pa^N(g_i)|$. Therefore, f_i assigns a state to gene g_i in dependence of the states of the parents of g_i in N. If the set of parents of g_i is empty, f_i becomes the empty set which is to be interpreted as some constant expression state of gene g_i. Usually non-redundancy of the functions f_i is demanded in the sense that for every parent $k \in \{1, \ldots, q_i\}$ of a gene g_i, there is a tuple $(b_1, \ldots, b_{q_i}) \in \mathbb{B}^{q_i}$ with $f_i(b_1, \ldots, b_{k-1}, 0, b_{k+1}, \ldots, b_{q_i}) \neq f_i(b_1, \ldots, b_{k-1}, 1, b_{k+1}, \ldots, b_{q_i})$.

The Boolean functions can be applied to the data values in two different ways, depending on the type of the data. In the case of time-course data, a natural way to apply a function f_i is to look up the values of the parents of g_i at time point t, apply f_i and compare the resulting state $s \in \mathbb{B}$ with the expression state of gene g_i at time point $t + 1$. In the case of gene disruption/overexpression data, f_i would be applied to the values of the parents in an experiment $v^{(j)}$, and the result would be compared to $v_i^{(j)}$.

If (N, F) is a good model of the real gene network, a good agreement, or even a perfect agreement of the function values to the data D is expected. This implies a natural way of assessing the quality of Boolean network models.

The above formulation allows general Boolean functions. However, researchers occasionally restrict to special kinds of Boolean functions. Several algorithms for the estimation of Boolean networks from discretised expression data have been proposed [Akutsu00b, Liang98].

2.2.2 Differential Equations

The modelling of gene networks with differential equations does in contrast to Boolean network models not require discretisation of the data, but is restricted to time-course data [Chen99, D'haeseleer99, Holter01, deHoon03a, Someren00]. Therefore, we assume we are given a data set of m vectors $x_j \in \mathbb{R}^n$ and time points $t_j \in \mathbb{R}$ $(j = 1, \ldots, m)$, where n again is the number of genes and $t_j < t_{j+1}$ holds for all $1 \leq j < m$. Each vector x_j corresponds to the gene expression measurements at a time point t_j. A gene network model is defined as a pair (N, Λ) of a network $N = (G, E)$ and an $n \times n$ matrix Λ with the following consistency condition: $\Lambda_{ij} \neq 0$ if and only if $(g_j, g_i) \in E$. The assumption made when using differential equations to describe gene networks is that for the true model the equation

$$\frac{d}{dt}x(t) = \Lambda \cdot x(t) \tag{2.1}$$

holds, where $x(t)$ denotes the gene expression values at an arbitrary point of time. We observe that this implies that the coaction of several genes during the regulation of a common target gene is modelled in an additive way.

We first introduce a score function based on this modelling approach in 2.2.2.1, then discuss our evaluation results based on synthetic data in 2.2.2.2.

2.2.2.1 A Score for Differential Equation Models

In the following, we describe the approach from [deHoon03a] to score these gene network models. In order to find an approximate solution for Equation 2.1 the differential was replaced by a difference equation in [Chen99] and in [deHoon03a] yielding

$$\frac{\Delta x}{\Delta t} = \Lambda \cdot x,$$

which can be rewritten as

$$x_{j+1} - x_j = (t_{j+1} - t_j) \cdot \Lambda \cdot x_j$$

for given data. This means a linear modelling of gene interaction. Since this is still an idealised modelling that does not account for errors in the data, we have to add an error ε_j for each time point:

$$x_{j+1} - x_j = (t_{j+1} - t_j) \cdot \Lambda \cdot x_j + \varepsilon_j$$

We assume that the error is time-independent and follows a normal distribution with mean 0 and some variance σ^2. Therefore, we expect the density of the error distribution to be

$$f(\varepsilon_j|\sigma^2) = (\frac{1}{\sqrt{2\pi\sigma^2}})^n exp\left(-\frac{\varepsilon_j^T \cdot \varepsilon_j}{2\sigma^2}\right),$$

where we assume an equal variance of the error σ^2 for all genes g_i. Taking the product over all time points j and applying the logarithm yields the log-likelihood of a given model (N, Λ):

$$L(\Lambda, \sigma^2) = -\frac{m \cdot n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \cdot \sum_{j=2}^{m} \hat{\varepsilon}_j^T \cdot \hat{\varepsilon}_j, \tag{2.2}$$

where $\hat{\varepsilon}_j$ denotes the error in the case that Λ is the true model:

$$\hat{\varepsilon}_j = x_{j+1} - x_j - (t_{j+1} - t_j) \cdot \Lambda \cdot x_j \tag{2.3}$$

In order to choose the variance σ^2 such that the likelihood is maximised, we look for a null of the derivative of the term in Equation 2.2 with respect to σ^2:

$$0 = -\frac{m \cdot n}{2\sigma^2} + \frac{1}{2\sigma^4} \cdot \sum_{j=2}^{m} \hat{\varepsilon}_j^T \cdot \hat{\varepsilon}_j$$

Therefore, the variance $\hat{\sigma}^2$ that maximises the likelihood is

$$\hat{\sigma}^2 = \frac{1}{m \cdot n} \cdot \sum_{j=2}^{m} \hat{\varepsilon}_j^T \cdot \hat{\varepsilon}_j. \tag{2.4}$$

By inserting this choice of the variance in Equation 2.2, we obtain

$$L(\Lambda, \hat{\sigma}^2) = -\frac{m \cdot n}{2} \log(2\pi\hat{\sigma}^2) - \frac{m \cdot n}{2}. \tag{2.5}$$

In the second step of maximisation of the log-likelihood, we need to find $\hat{\Lambda}$ that maximises $L(\Lambda, \hat{\sigma}^2)$. As we see from Equation 2.5, maximising the log-likelihood means to minimise $\hat{\sigma}^2$, which by Equation 2.3 and Equation 2.4 can be written as

$$\begin{aligned} \hat{\sigma}^2 = \frac{1}{m \cdot n} \sum_{j=2}^{m} \; & ((x_j^T - x_{j-1}^T) \cdot (x_j - x_{j-1}) \\ & + (t_j - t_{j-1})^2 \cdot x_{j-1}^T \cdot \Lambda^T \cdot \Lambda \cdot x_{j-1} \\ & - 2 \cdot (x_j^T - (t_j - t_{j-1}) \cdot x_{j-1}^T) \cdot \Lambda \cdot x_{j-1}). \end{aligned}$$

Taking the derivative in Λ yields [deHoon03a]

$$\hat{\Lambda} = B \cdot A^{-1}, \tag{2.6}$$

where the matrices A and B are defined as:

$$\begin{aligned} A &= \sum_{j=2}^{m} (t_j - t_{j-1})^2 \cdot x_{j-1} \cdot x_{j-1}^T \\ B &= \sum_{j=2}^{m} (t_j - t_{j-1}) \cdot (x_j - x_{j-1}) \cdot x_{j-1}^T \end{aligned} \tag{2.7}$$

Therefore, when neglecting the above restriction that matrix entries not corresponding to an edge in the network N must be zero, Λ can be calculated directly. Now, in order to find a good trade-off of model fitting and model complexity, we make use of the Akaike information criteria [Akaike73]

$$AIC = -2 \cdot [\text{log-likelihood}] + 2 \cdot p, \tag{2.8}$$

which can be used as the score of a given gene network model. Here, p denotes the number of parameters of the model, which in our case is the number of edges of the network plus one. In [deHoon03a], it was shown that the derivation above can be done in a similar way, if some matrix entries are restricted to be zero by a given mask M, which is a $n \times n$ matrix with entries in $\mathbb{B}$. Therefore, for a given

mask the score can be computed efficiently and search strategies can be applied to find a mask (a network) with a good score.
An alternative to the AIC score is the BIC score [Schwarz78]:

$$BIC = -2 \cdot [\text{log-likelihood}] + p \cdot \log m, \tag{2.9}$$

We note that in [deHoon03b] it was shown that the problem of searching an optimal mask can be decomposed to searching parents for each gene independently, if the variance of the error is modelled as different for each gene.

2.2.2.2 Evaluations with Synthetic Data

Gene expression measurements are subject to error. The source of error can be the process of measuring or the variability of the biological system itself. These two different types of error are called *measurement error* and *system error*, respectively. We have evaluated the influence of the error on the accuracy of estimations based on the score function described above in a computational experiment. In the second computational experiment, we evaluated the effect of a varying number of measurements.

We designed artificial networks and used them to generate synthetic expression data. We then applied an exhaustive search strategy (see 2.4.2.2) to the simulated data and compared the estimated network to the true network.

The design of the networks was done as follows. First, we only considered sparse networks, since actual gene networks are assumed to be sparse. Second, we defined the matrix entries in such a way that all eigenvalues of the matrix Λ have negative real parts. Consequently, the true network is stable meaning that the expression values do not grow without bound. Since stability is a fundamental property of biological systems, this restriction seems to be important.

When generating synthetic gene expression data from these networks, we added an error. The error was sampled from a normal distribution with mean 0 and variance σ^2. We focused on a single edge of the synthetic gene network and checked whether it is included in the estimated network. For a fixed magnitude of measurement error, we repeated this 400 times and computed the proportion of estimations including the edge under consideration. We then repeated this procedure for varying magnitudes of measurement error. The result is shown in Figure 2.1. As expected, the reliability decreases for increasing measurement error. If the measurement error is significantly larger than the magnitude of the gene expression levels, the estimation reliability decreases rapidly.

In our second evaluation, we measured the estimation accuracy by counting the number of incorrectly estimated edges (false positives as well as false negatives), not including self-regulatory effects. By repeating the data generation and

network estimation many times, we estimated the expected value of the number of incorrect edges. In Figure 2.2, we show the relationship of the number of measurements and the estimation accuracy. As expected, the estimation accuracy improves as the number of measurements increases. We also observe that the BIC performs significantly better than the AIC. As a result of this evaluation, it does not seem to be advisable to do gene network estimations from much less than about 20 measurements. In contrast, using about 50 measurements should allow for gene network estimations that agree with the target network to a high degree, if the linear modelling is appropriate for real gene networks. However, since we used a synthetic network of six genes for this evaluation, this only holds for very small networks. For larger networks, a higher number of measurements might be necessary.

2.2.3 Bayesian Networks

Bayesian networks [Cooper92, Friedman98b, Friedman00, Hartemink01, Imoto02, Imoto03b, Ong02, Pe'er01, V.A.Smith02] are a widely used approach to the modelling of gene networks. While gene regulation is modelled by Boolean functions in Boolean networks, and as linear functions in differential equation models, probability distributions are used in Bayesian networks. We first introduce the general concept of static Bayesian networks, and then mention the special case of dynamic Bayesian networks.

2.2.3.1 Static Bayesian Networks

In Bayesian network models, the expression of a gene g_i is modelled as a random variable X_i $(i = 1, \ldots, n)$. The behaviour of the gene network is modelled as a joint probability distribution of the gene expression patterns for all genes. A Bayesian network model of a gene network is a pair (N, P) of an directed acyclic graph N and a joint probability distribution P that fulfils the following condition. The joint probability distribution can be decomposed as a product of conditional probabilities $P^{(i)}(X_i|X_{k_1}, \ldots, X_{k_q})$ $(q = |Pa^N(g_i)|)$.

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P^{(i)}(X_i|Pa^N(X_i)) \tag{2.10}$$

Here, we use $Pa^N(X_i)$ in accordance to $Pa^N(g_i)$ (Definition 4). Each conditional probability distribution $P^{(i)}$ models the regulation of a gene g_i by some other genes $g_{k_1}, \ldots, g_{k_q}$. Since the conditional probability distributions depend only on the parents of genes in a given graph N, Bayesian networks are based on the Markov assumption that there is an acyclic network such that the expression of

Figure 2.1: The estimation reliability as a function of the measurement error. For each noise over signal ratio, the proportion of correct predictions was calculated from a total of 400 runs. In each run, 100 synthetic data points were generated, from which the gene network was estimated.

Figure 2.2: The estimation accuracy as a function of the number of measurements. An artificial network of six genes was used.

a gene g_i depends only on the parents of g_i in N. We are restricted to acyclic networks, because Equation 2.10 does not hold for cyclic networks N. However, the restriction of acyclicity refers only to directed cycles, but not to undirected cycles. Therefore, N may contain undirected cycles and can have any structure, when the directions of the edges are omitted.

Modelling gene regulation by probability distributions allows for a very general modelling of gene interactions. Depending on the concrete probability distribution chosen in applications, even non-linear gene interactions can be modelled.

Scoring Bayesian network models is usually based on the posterior probability of the model in the light of given data. Examples for this approach are the BDe score and the BNRC score introduced in Section 2.3. The general scoring scheme is

$$\begin{aligned} score(N) &\propto P(N|Data) \\ &= \frac{P(Data|N) \cdot P(N)}{P(Data)}, \end{aligned} \tag{2.11}$$

where the second term follows by the Bayes theorem. Since actual probabilities might be very small numbers in practice, usually the logarithm of the posterior probability is used instead in order to avoid arithmetic underflow.

$$\begin{aligned} score(N) &= -\log P(N|Data) \\ &= -\log P(Data|N) - \log P(N) + \log P(Data) \\ &= -\log P(Data|N) - \log P(N) + const \end{aligned} \tag{2.12}$$

We observe that minimising a score of this type is equivalent to maximising the posterior probability. The term $P(Data|N)$ is called the *marginal likelihood*, $P(N)$ the *prior probability* of the network. Concrete examples for these probability distributions will be given in Section 2.3. The term $\log P(Data)$ is invariant with respect to the selected gene network model, and, therefore, does not need to be considered in order to find the most likely model.

2.2.3.2 Dynamic Bayesian Networks

In some cases, gene expression data is given as time-course data, providing gene expression measurements for all genes at some given time points. In this case, a special kind of Bayesian networks can be applied, which are called dynamic Bayesian networks [Friedman98a, Kim03, Ong02, V.A.Smith02, V.A.Smith03]. Similar to the case of differential equation models, one attempts to explain the gene expression pattern at a time point t_j from the observed gene expression pattern at time point t_{j-1}. Therefore, these networks can be modelled using a bipartite graph that has two nodes for each gene, one node representing the gene

at a preceding time point, the other node representing the node at the following time point. Since gene expression is explained from preceding time points, such networks are necessarily bipartite and also acyclic. This expanded network used for network estimation can be reduced to a graph on G by merging the pair of nodes for every gene. The merged network can be of any structure, even directed cycles are possible.

This approach also allows to search for parents of a gene independently of other genes. However, since it makes intrinsic use of the time dimension, this approach is restricted to time-course data and can not be applied to general data as, for example, data obtained from gene disruption/overexpression experiments.

Because there is a close relationship of differential equation models and dynamic Bayesian networks [deHoon03b], we also use the term *dynamic networks* as a generic term for both approaches.

2.3 Score Functions for Bayesian Networks

In order to evaluate different gene network models, we need to define score functions. In this section, we introduce three score functions for the Bayesian network framework that have been applied for research: the MDL score (Subsection 2.3.1), the BDe score (Subsection 2.3.2), and the BNRC score (Subsection 2.3.3). Furthermore, we compare the estimation accuracy of the BNRC score and the score for differential equations defined above, in application to real data (Subsection 2.3.4). Finally, we address the different possibilities to define meaningful prior probability distributions in Subsection 2.3.5.

As in the previous section, we assume we are given genes $G = \{g_1, \ldots, g_n\}$ and model each gene g_i by a random variable X_i.

2.3.1 MDL score

The MDL score [Friedman98b] uses discretised data and makes use of the minimal description length (MDL) principle. The idea of the MDL principle [Rissanen89] is that if a structure in some given data D can be detected, this structure (also called model) can be used to construct a compressed description D' of D. Since in order to decode D' the model is necessary, D' has to be stored together with the model in order to have a compressed description of D that can be decoded. The total description length of D is, therefore, defined as the description length of D' plus the description length of the model. When a space of models under consideration is given, the MDL principle dictates to choose the model that minimises the total description length. The MDL principle, therefore, provides a balancing

of the complexity of the model and the accuracy of the description of the data using the model.

We now introduce a score function from [Friedman98b] that applies the MDL principle to Bayesian networks. In the case of Bayesian networks, the data is gene expression data and the models under consideration are joint probability distributions of a certain kind as described below. A joint probability distribution can be used to compress gene expression data by assigning shorter descriptions to gene expression patterns with higher probability.

The gene expression data is supposed to be discretised. Therefore, we assume we are given a finite set of possible values $Val(X_i)$ for each random variable X_i. For sets of random variables $A \subseteq \{X_1, \ldots, X_n\}$, we use $Val(A)$ to denote the space of possible combinations of values for the random variables in A. We also assume we are given a data set $D = u_1, \ldots, u_p$ of gene expression patterns $u_i \in Val(\{X_1, \ldots, X_n\})$.

The models are joint probability distributions that are decomposed by a network $N = (G, E)$ to conditional probability distributions (CPDs) for each gene. Therefore, if we are given a network, all we need to specify in order to define a joint probability distribution are probabilities $\theta_{x_i|pa_i}$ for all $x_i \in Val(X_i)$ and all $pa_i \in Val(Pa^N(X_i))$, such that $\sum_{x_i \in Val(X_i)} \theta_{x_i|pa_i} = 1$ for all $i = 1, \ldots, n$ and all $pa_i \in Val(Pa^N(X_i))$. Denoting a set of probabilities $\theta_{x_i|pa_i}$ for all X_i as $\mathcal{L}$, a joint probability distribution is specified by a pair $B = (N, \mathcal{L})$.

We now define the description length of a given model $B = (N, \mathcal{L})$. First, we can describe $N = (G, E)$ by encoding the parents $Pa^N(X_i)$ for each $i = 1, \ldots, n$. This requires $\log_2 n$ bits for encoding the number of parents $|Pa^N(X_i)|$ for each gene, and

$$\log_2 \binom{n}{|Pa^N(X_i)|}$$

bits for storing an index of the subset $Pa^N(X_i)$. Therefore, the description length for N can be defined as

$$DL_N = \sum_{i=1}^{n} (\log_2 n + \log \binom{n}{|Pa^N(X_i)|}). \tag{2.13}$$

We note that the description length for N becomes highest for $|Pa^N(X_i)| = \lceil \frac{n}{2} \rceil$ and $|Pa^N(X_i)| = \lfloor \frac{n}{2} \rfloor$, but lowest for $|Pa^N(X_i)| = 0$ and $|Pa^N(X_i)| = n$. Therefore, when the number of parents for a given gene g_i increases, the description length increases at first but declines again after the number of parents exceeded $\lceil \frac{n}{2} \rceil$. While the increase is biologically plausible, the decrease for high numbers of parents is not. Therefore, when the MDL score is used selection of more than $\lceil \frac{n}{2} \rceil$ parents should be suppressed.

Next, we need to define the description length of $\mathcal{L}$. The description length for one probability in a CPD table is usually chosen as $\frac{1}{2} \cdot \log_2 p$ [Friedman96]. Therefore, the description length for $\mathcal{L}$ can be defined as

$$DL_{\mathcal{L}} = \sum_{i=1}^{n} \frac{1}{2} \cdot |Val(Pa^N(X_i))| \cdot (|Val(X_i)| - 1) \cdot \log_2 \ p. \tag{2.14}$$

Combining Equation 2.13 and Equation 2.14, the description length of $B = (N, \mathcal{L})$ becomes

$$DL_B = DL_N + DL_{\mathcal{L}}.$$

Given a model B, the probability distribution P_B specified by B can be used to encode the data D defined above. Since the given gene expression patterns $u_i \in Val(\{X_1, \ldots, X_n\})$ with a high probability $P_B(u_i)$ are expected to appear more often in the data, it is reasonable to select short codes to encode these. An example for a coding of this type is the Huffman code [Cover91]. The optimal encoding length for a gene expression pattern u_i in the Huffman code can be approximated by $-\log P_B(u_i)$ [Cover91]. Therefore, we define the description length of the data as:

$$DL_D(B) = -\sum_{i=1}^{p} \log P_B(u_i) \tag{2.15}$$

The total description length is the description length of the model plus the description length of the encoded data using the model.[3]

Definition 11: [Friedman98b]
Let D be a discretised data set and $B = (N, \mathcal{L})$ be a specification of a gene network model. We define the *MDL score* as

$$score_{\text{MDL}}(B) = DL_B + DL_D(B) \qquad \bullet$$

In order to compute the MDL score we need to compute $\mathcal{L}$ for given networks N. It can be shown [Friedman98b] that the CPDs that minimise $DL_D(B)$ are equal to the observed frequencies of gene expression patterns in the data. Therefore, in order to find the optimal setting for $\theta_{x_i|pa_i}$ ($i \in \{1, \ldots, n\}$, $x_i \in Val(X_i)$, $pa_i \in Val(Pa^N(X_i))$), we can count the number $N(pa_i)$ of observed gene expression patterns that match the values of pa_i for the parents of X_i, then count the number $N(x_i, pa_i)$ among these patterns that also show the value x_i for X_i, and set

$$\theta_{x_i|pa_i} = \begin{cases} \frac{N(x_i, pa_i)}{N(pa_i)} & \text{if } |Pa^N(X_i)| > 0, \\ \frac{N(x_i)}{p} & \text{if } |Pa^N(X_i)| = 0. \end{cases}$$

[3] We can omit the description of the number of genes and the size of $Val(X_i)$ for each gene, because this description length is independent of the selected model.

In this work, we make use of an implementation of the MDL score that follows Definition 11.

2.3.2 BDe score

The BDe score[4] [Buntine91, Cooper92, Friedman98b, Heckerman95] is defined as proportional to the posterior probability of the network, given the data. When the BDe score is used, the microarray data needs to be discretised as in the case of the MDL score.

Given discretised data D as defined in the previous subsection, possible values $Val(X_i)$ for every random variable X_i, and a network N, the posterior probability of the network is by Equation 2.11

$$P(N|D) = \alpha \cdot P(D|N) \cdot P(N).$$

In this work, we set the prior probability $P(N)$ similar to [Friedman98b] as

$$P(N) \propto \frac{1}{2^{DL_N}} \tag{2.16}$$

for a given graph N with DL_N as defined in Equation 2.13. To compute the marginal likelihood $P(D|N)$, we have to consider all possible parameter settings Θ_N for the CPDs, and have to compute

$$P(D|N) = \int P(D|\Theta_N, N) \cdot P(\Theta_N|N) d\Theta_N. \tag{2.17}$$

Assuming that each CPD can be learned independently from the other CPDs, this integral can be decomposed to the following term, where we use $N(x_i, pa_i)$ as in Subsection 2.3.1 to denote the number of gene expression patterns in D that match x_i and pa_i [Cooper92, Heckerman95]:

$$P(D|N) = \prod_i \prod_{\substack{pa_i \in \\ Val(Pa^N(X_i))}} \int \prod_{x_i \in Val(X_i)} \theta_{x_i|pa_i}^{N(x_i,pa_i)} \cdot P(\Theta_{X_i|pa_i}|N) d\Theta_{X_i|pa_i} \tag{2.18}$$

We need to select a prior probability of the CPD parameters $\Theta_{X_i|pa_i}$. In the case of the BDe score a Dirichlet prior [DeGroot70] is chosen, which allows the following closed-form solution of the integral in Equation 2.18. The hyperparameters of the Dirichlet prior are denoted as $N'_{x_i|pa_i}$ and Γ denotes the gamma function (see Definition 10).

$$P(D|N) = \prod_i \prod_{pa_i} \frac{\Gamma(\sum_{x_i} N'_{x_i|pa_i})}{\Gamma(\sum_{x_i} N'_{x_i|pa_i} + N(pa_i))} \prod_{x_i} \frac{\Gamma(N'_{x_i|pa_i} + N(x_i, pa_i))}{\Gamma(N'_{x_i|pa_i})} \tag{2.19}$$

[4] BD stands for *B*ayesian *D*irichlet, and BDe abbreviates likelihood-*e*quivalent BD metric.

In [Heckerman95] the following selection of hyperparameters for the Dirichlet prior was proposed. A prior model $B_p = (N_p, \mathcal{L}_p)$ and an equivalent sample size p' is selected, and $N'_{x_i|pa_i}$ is set to $p' \cdot P_{B_p}(x_i, pa_i)$.
Using above equations, we can now define the BDe score.

Definition 12: [Heckerman95]
Let D be a discretised data set and N be a directed acyclic graph. With $P(D|N)$ as defined in Equation 2.19 and $P(N)$ as in Equation 2.16, we define the *BDe score* as

$$score_{\text{BDe}}(N) = \log P(D|N) + \log P(N) \qquad \bullet$$

For large data sets, the MDL score and the BDe score have been shown to be asymptotically equivalent [Bouckaert94, Schwarz78]. However, since the number of microarrays in available data sets is small, the two scores may show a different behaviour in practice.

In this work, we make use of an implementation of the BDe score that follows Definition 12.

2.3.3 BNRC score

The BNRC score[5] [Imoto02] is, as the BDe score, defined as proportional to the posterior likelihood of the network, given the data. But in contrast to the BDe score a continuous probability distribution is used, therefore making discretisation of the expression data unnecessary. Also, the nonparametric regression that is applied to the given data points allows for non-linear gene interactions. We first give the definition of the continuous probability distribution used by the BNRC score, then give the general definition of the score, and finally give details of the concretisation of the BNRC score as used for the evaluations in this thesis.

2.3.3.1 Continuous Probability Distribution

We assume we are given a data set in the form of a $p \times n$ matrix D, where p denotes the number of microarrays and n the number of genes. Let N be a directed acyclic graph. Let us use q_i to denote the number of parents of a gene g_i in N, that is $q_i = |Pa^N(g_i)|$ $(i = 1, \ldots, n)$. We choose an arbitrary sorting for the elements of every set of parents $Pa^N(g_i)$ and use $p_{jk}^{(i)}$ to denote the expression value of the k-th parent of g_i in the j-th row of D (i.e. the j-th microarray), $i = 1, \ldots, n$, $j = 1, \ldots, p$, $k = 1, \ldots, q_i$. We also use p_{ji} to denote the vector

[5] BNRC abbreviates *B*ayesian Network and *N*onparametric *R*egression *C*riterion.

$(p_{j1}^{(i)}, \ldots, p_{jq_i}^{(i)})^T$. Using these notations, we can reformulate Equation 2.10 using density functions instead of probability distributions.

$$f(x_{j1}, x_{j2}, \ldots, x_{jn}) = f_1(x_{j1}|p_{j1}) \cdot f_2(x_{j2}|p_{j2}) \cdot \ldots \cdot f_n(x_{jn}|p_{jn}). \tag{2.20}$$

Now all we need to do in order to define a continuous probability distribution for the continuous gene expression patterns is to define the conditional densities $f_i(x_{ji}|p_{ji})$ $(i = 1, \ldots, n)$. The BNRC score fits smooth functions to the given gene expression data:

$$\begin{aligned} &x_{ji} = m_1(p_{j1}^{(i)}) + m_2(p_{j2}^{(i)}) + \ldots + m_{q_i}(p_{jq_i}^{(i)}) + \varepsilon_{ji} \\ &j = 1, \ldots, p;\ i = 1, \ldots, n \end{aligned} \tag{2.21}$$

Here, $m_k : \mathbb{R} \to \mathbb{R}$ $(k = 1, \ldots, q_i)$ are smooth functions and the ε_{ji} are assumed to be normally distributed with variance σ_i^2. This regression approach allows to model the interaction of a pair of genes in a very general, non-linear way. We note, however, that the coaction of several parent genes during the regulation of the target gene g_i is modelled in an additive way. Though this is supposed to capture many aspects of gene regulation, it may potentially limit the estimation accuracy in cases where gene regulation phenomena deviate substantially from additivity.

The structure of function m_k is assumed to be as given in the following equation:

$$m_k(p_{jk}^{(i)}) = \sum_{m=1}^{M} \gamma_{mk}^{(i)} \cdot b_{mk}^{(i)}(p_{jk}^{(i)}), \quad j = 1, \ldots, p;\ k = 1, \ldots, q_i \tag{2.22}$$

$b_{1k}^{(i)}, \ldots, b_{Mk}^{(i)}$ denote basis functions which can be, for example, polynomial functions or spline functions. The concrete definition of the basis functions as used in the evaluations of this work is given in 2.3.3.3. M is the number of basis functions and $\gamma_{1k}^{(i)}, \ldots, \gamma_{Mk}^{(i)}$ are parameters, over which we will later integrate similar to the case of the BDe score (Equation 2.17).

Using the nonparametric regression, we can now define the conditional density functions in the following way.

$$f_i(x_{ji}|p_{ji}; \gamma_i; \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[-\frac{\{x_{ji} - \sum_{k=1}^{q_i} \sum_{m=1}^{M} \gamma_{mk}^{(i)} \cdot b_{mk}^{(i)}(p_{jk}^{(i)})\}^2}{2\sigma_i^2} \right] \tag{2.23}$$

Here, we use γ_i to abbreviate $(\gamma_{i1}^T, \ldots, \gamma_{iq_i}^T)^T$, where γ_{ik} denotes $(\gamma_{1k}^{(i)}, \ldots, \gamma_{Mk}^{(i)})^T$. In the case of genes without parents $(q_i = 0)$, the expression levels of the genes are modelled by a normal distribution with mean μ_i and variance σ_i^2.

Putting everything together, we can now give an expression for a gene network model in the sense of the BNRC score. We use θ_N to denote the vector of all parameters that are necessary to specify a joint probability distribution for the given network structure N. Therefore θ_N is $(\theta_1^T, \ldots, \theta_n^T)^T$, where θ_i is the vector of all parameters for gene g_i:

$$\theta_i = \begin{cases} (\gamma_i^T, \sigma_i^2)^T & \text{if } q_i > 0, \\ (\mu_i, \sigma_i^2)^T & \text{if } q_i = 0. \end{cases} \tag{2.24}$$

Using above notations, a gene network model in the sense of the BNRC score can be written as

$$f(x_j; \theta_N) = \prod_{i=1}^{n} f_i(x_{ji} | p_{ji}; \theta_i), \quad j = 1, \ldots, p.$$

2.3.3.2 General Score Definition

As in the case of the BDe score (see Equation 2.17), the BNRC score is defined as proportional to the posterior likelihood of the network N, given the data. This likelihood can be computed by integrating over all possible parameterisations for joint probability distributions based on N. We denote the prior distribution of the unknown parameter vector θ_N as $\pi(\theta_N|\lambda)$, where λ is the matrix of hyper parameters (some hyper parameters for each gene). The hyper parameters will be used to regulate the smoothness respectively the grossness of curve fitting when applying the regression from Equation 2.21. We assume $\log \pi(\theta_N|\lambda) = O(p)$.

$$P(N|D) \propto \pi_N \cdot \int \prod_{j=1}^{p} f(x_j; \theta_N) \cdot \pi(\theta_N|\lambda) d\theta_N \tag{2.25}$$

Here, π_N is the prior probability of N. In this work, we set π_N as

$$\begin{aligned} \pi_N &\propto \prod_{i=1}^{n} \pi_N^{(i)} \\ &= \prod_{i=1}^{n} \frac{1}{p^{|Pa^N(g_i)|}} \end{aligned} \tag{2.26}$$

Since the prior probability reduces sharply for an increasing number of parent genes, the prior probability is used to avoid overfitting in a similar way to the case of the AIC (Equation 2.8) and the BIC (Equation 2.9).

To compute the integral in Equation 2.25 we use the Laplace approximation

for integrals [Davison86, Heckerman98, Tinerey86] which yields

$$\begin{aligned}\int \prod_{j=1}^{p} f(x_j;\theta_N)\cdot\pi(\theta_N|\lambda)d\theta_N &= \int \exp\{p\cdot l_\lambda(\theta_N|D)\}d\theta_N \\ &= \frac{(2\pi/p)^{r/2}}{|J_\lambda(\hat{\theta}_N)|^{1/2}}\cdot\exp\{p\cdot l_\lambda(\hat{\theta}_N|D)\}\cdot\{1+O(p^{-1})\},\end{aligned} \tag{2.27}$$

where r is the dimension of θ_N,

$$\begin{aligned} l_\lambda(\theta_N|D) &= \frac{1}{p}\sum_{j=1}^{p}\log f(x_j;\theta_N)+\frac{1}{p}\cdot\log\pi(\theta_N|\lambda), \\ J_\lambda(\theta_N) &= -\partial^2\{l_\lambda(\theta_N|D)\}/\partial\theta_N\partial\theta_N^T, \end{aligned} \tag{2.28}$$

and $\hat{\theta}_N$ is the maximum point of $l_\lambda(\theta_N|D)$.

Definition 13: [Imoto02]
Let D be a data set and N be a directed acyclic graph. Using the terms defined in Equation 2.26 and Equation 2.28, we define

$$\begin{aligned} score_{\mathrm{BNRC}}(N) &= -2\cdot\log\left\{\pi_N\cdot\int\prod_{j=1}^{p}f(x_j;\theta_N)\cdot\pi(\theta_N|\lambda)d\theta_N\right\} \\ &= -2\cdot\log\pi_N - r\cdot\log(2\pi/p)+\log|J_\lambda(\hat{\theta}_N)|-2p\cdot l_\lambda(\hat{\theta}_N|D). \end{aligned}$$ •

We also state the decomposition of the BNRC score in scores for the single genes. This decomposition is based on the assumption that $\pi(\theta_N|\lambda)$ can be decomposed to

$$\pi_N(\theta_N|\lambda)=\pi_1(\theta_1|\lambda_1)\cdot\ldots\cdot\pi_n(\theta_n|\lambda_n).$$

Using this decomposition, $l_\lambda(\hat{\theta}_N|D)$ and $\log|J_\lambda(\hat{\theta}_N)|$ in Definition 13 are, respectively,

$$\sum_{i=1}^{n} l_\lambda^{(i)}(\hat{\theta}_i|D) \quad\text{and}\quad \sum_{i=1}^{n}\log\left|-\frac{\partial^2 l_\lambda^{(i)}(\theta_i|D)}{\partial\theta_i\partial\theta_i^T}\right|,$$

where

$$l_\lambda^{(i)}(\theta_i|D)=\frac{1}{p}\cdot\sum_{j=1}^{p}\log f_i(x_{ji}|p_{ji};\theta_i)+\frac{1}{p}\cdot\log\pi_i(\theta_i|\lambda_i). \tag{2.29}$$

With $\pi_N^{(i)}$ as defined in Equation 2.26, we can write the local scores for genes g_i as

$$score_{\text{BNRC}}^{(i)}(N) = -2 \cdot \log \left\{ \pi_N^{(i)} \cdot \int \prod_{j=1}^{p} f_i(x_{ji}|p_{ji};\theta_i) \cdot \pi_i(\theta_i|\lambda_i) d\theta_i \right\}, \tag{2.30}$$

and obtain

$$score_{\text{BNRC}}(N) = \sum_{i=1}^{n} score_{\text{BNRC}}^{(i)}(N). \tag{2.31}$$

2.3.3.3 Concretisation

In order to completely specify the BNRC score we need to define the basis functions (Equation 2.22) and the prior distribution of the parameters θ_i (Equation 2.24). The basis functions are selected as M B-splines [deBoor78] of degree 3 placed at equal distances in the domain $[\min_j(p_{jk}^{(i)}), \max_j(p_{jk}^{(i)})]$ which is divided in $M+3$ intervals. For the prior distribution, we assume

$$\pi_i(\theta_i|\lambda_i) = \prod_{k=1}^{q_i} \pi_{ik}(\gamma_{ik}|\lambda_{ik}),$$

where $\pi_{ik}(\gamma_{ik}|\lambda_{ik})$ is an M-variate normal distribution given by

$$\pi_{ik}(\gamma_{ik}|\lambda_{ik}) = \left(\frac{2\pi}{p \cdot \lambda_{ik}}\right)^{-(M-2)/2} \cdot |K_{ik}|_+^{1/2} \cdot \exp\left(-\frac{p \cdot \lambda_{ik}}{2} \cdot \gamma_{ik}^T K_{ik} \gamma_{ik}\right). \tag{2.32}$$

K_{ik} is an $M \times M$ matrix with

$$\gamma_{ik}^T K_{ik} \gamma_{ik} = \sum_{l=3}^{M} (\gamma_{lk}^{(i)} - 2 \cdot \gamma_{l-1,k}^{(i)} + \gamma_{l-2,k}^{(i)})^2,$$

and $|K_{ik}|_+$ is the product of $M-2$ nonzero eigenvalues of K_{ik}.
Having now fully specified the BNRC score, we obtain

$$\begin{aligned} score_{\text{BNRC}}^{(i)}(N) = &-2 \cdot \log \pi_N^{(i)} - 2 \cdot \sum_{j=1}^{p} \log f_i(x_{ji}|p_{ji};\hat{\theta}_i) - 2 \cdot \sum_{k=1}^{q_i} \log \pi_k(\hat{\gamma}_{ik}|\lambda_{ik}) \\ &+ \log \left| -\frac{\partial^2 l_\lambda^{(i)}(\hat{\theta}_i|D)}{\partial \theta_i \partial \theta_i^T} \right| - (\sum_{k=1}^{q_i} M + 1) \cdot \log(2\pi \cdot p^{-1}), \end{aligned} \tag{2.33}$$

where $\hat{\theta}_i = (\hat{\gamma}_i^T, \hat{\sigma}_i^2)^T$ maximises $l_\lambda^{(i)}(\theta_i|D)$. The logarithm of the determinant of the Hessian matrix in Equation 2.33 can be approximated as

$$\sum_{k=1}^{q_i} \{\log |B_{ik}^T B_{ik} + p\hat{\sigma}_i^2 \lambda_{ik} \cdot K_{ik}| - M \cdot \log(p \cdot \hat{\sigma}_i^2)\} - \log(2 \cdot \hat{\sigma}_i^4),$$

where B_{ik} denotes an $n \times M$ matrix defined by $B_{ik} = (b_{ik}(p_{1k}^{(i)}), \ldots, b_{ik}(p_{pk}^{(i)}))^T$ with $b_{ik}(p_{jk}^{(i)}) = (b_{1k}^{(i)}(p_{jk}^{(i)}), \ldots, b_{Mk}^{(i)}(p_{jk}^{(i)}))^T$. Combining Equations 2.23, 2.32, and 2.33, $score_{\text{BNRC}}^{(i)}(N)$ can be written as

$$\begin{aligned} score_{\text{BNRC}}^{(i)}(N) = {} & C_i + (p - 2 \cdot q_i - 2) \cdot \log \hat{\sigma}_i^2 \\ & + \sum_{k=1}^{q_i} \left\{ \frac{p \cdot \beta_{ik}}{\hat{\sigma}_i^2} \cdot \hat{\gamma}_{ik}^T K_{ik} \hat{\gamma}_{ik} + \log |\Lambda_{ik}| - (M-2) \cdot \log \beta_{ik} \right\}, \end{aligned} \tag{2.34}$$

where β_{ik} denotes $\sigma_i^2 \cdot \lambda_{ik}$, and

$$\begin{aligned} C_i &= -2 \cdot \log \pi_N^{(i)} + (p + q_i \cdot M - 2 \cdot q_i) \cdot \log(2\pi) + p - \log 2 \\ &\quad -2(q_i \cdot M - q_i) \cdot \log p - \sum_{k=1}^{q_i} \log |K_{ik}|_+ \\ \Lambda_{ik} &= B_{ik}^T B_{ik} + p\beta_{ik} \cdot K_{ik}. \end{aligned} \tag{2.35}$$

Finally, we can use the following algorithm to approximate $\hat{\theta}_i = (\hat{\gamma}_i^T, \hat{\sigma}_i^2)^T$ that maximises $l_\lambda^{(i)}(\theta_i|D)$ when the values of β_{ik} are given.

Algorithm 1: Backfitting Algorithm [Hastie90]

Step 1:	Initialize: $\gamma_{ik} = \mathbf{0}$, $k = 1, \ldots, q_i$.
Step 2:	For all $k = 1, \ldots, q_i$, do the following step.
Step 2a:	$\gamma_{ik} = (B_{ik}^T B_{ik} + p\beta_{ik} \cdot K_{ik})^{-1} B_{ik}^T (x_{(i)} - \sum_{k' \neq k} B_{ik'} \gamma_{ik'})$.
Step 3:	Continue Step 2 until a suitable convergence criterion is satisfied.
Step 4:	Compute $\hat{\sigma}_i^2$ as $\|\|x_{(i)} - \sum_{k=1}^{q_i} B_{ik} \hat{\gamma}_{ik}\|\|^2 / p$.

The selection of values for β_{ik} is done by choosing one out of several values that minimises the BNRC score.

In this work, we make use of an implementation of the BNRC score that follows Definition 13.

2.3.4 Comparison of BNRC and Differential Equations

In order to compare the estimation accuracy of the BNRC score and the score for differential equation models described in 2.2.2.1, we selected a set of 6 sigma

factors of *Bacillus subtilis* and 80 operons known to be regulated by one of the sigma factors [Sonenshein01]. For each operon, we then composed a set of 7 genes containing all sigma factors and the operon. For all 80 sets, we estimated optimal network models with respect to the BNRC score as well as optimal network models with respect to the score for differential equation models.

The microarray data used for this evaluation is a set of 65 [6] microarrays obtained from time-course experiments for *B. subtilis* (introduced in more detail in Subsection 4.3.1). The expression measurements of the genes within the same operon were averaged.

In the case of the BNRC score, we made use of Algorithm 3 for the optimal estimations, which will be introduced in Chapter 3. We restricted the number of parents that a gene may have to one. In the case of differential equations, we applied an exhaustive search strategy from [deHoon03b] and restricted the number of parents to at most two, since in the case of differential equations a gene is estimated as one of its parents in almost all estimations. Therefore, restricting the number of parents to two corresponds to restricting the number of different parents to one. We then checked, whether the regulatory relation of the correct sigma factor and the operon was included in the estimation. Table 2.1 summarises the result of 80 gene network estimations for both scores.

Sigma Factor	Operons in Regulon	Correct Estimations Diff. Equations	p-value Differential Equations	Correct Estimations BNRC	p-value BNRC
$sigE$	22	4	0.51	1	0.99
$sigD$	16	1	0.95	12	$3.81 \cdot 10^{-5}$
$sigW$	21	18	$7.8 \cdot 10^{-12}$	18	$8.61 \cdot 10^{-9}$
$sigH$	11	2	0.57	9	$1.26 \cdot 10^{-4}$
$sigX$	6	3	0.062	5	$4.63 \cdot 10^{-3}$
$sigF$	4	1	0.52	3	0.05

Table 2.1: Comparison of the estimation accuracy of the BNRC and differential equations.

The table shows the number of correctly estimated relations. In the case of differential equations, the regulatory relation was counted as detected, if the parent of the operon that was different from the operon matched the sigma factor. In the case of Bayesian networks, we checked for an undirected edge connecting the sigma factor and the operon. As discussed in Subsection 2.2.3, checking the undirected structure of the estimation result instead of the directed structure is

[6] From the data set, the microarrays from experiments under glucose limitation were excluded.

advisable, because in the case of directed structures the result might be influenced by the acyclicity of the network.

The p-value for correctly estimating the regulatory sigma factor for k out of n operons were computed by applying Bernoulli's formula

$$P(k,n) = \sum_{i=k}^{n} \binom{n}{i} \cdot p^i \cdot (1-p)^{n-i}, \tag{2.36}$$

where p denotes the probability of guessing the sigma factor correctly. This value was determined as $\frac{1}{6}$ in the case of the differential equation models, since only one out six sigma factors can be a parent of the operon. In the case of Bayesian networks, p is $\frac{1}{4}$ as we determined by enumerating all acyclic networks with 7 genes with at most one parent for each node, and counting the fraction of networks that include a fixed, undirected edge.

From the result in Table 2.1, we observe that the estimations using differential equations were strongly significant in one case ($sigW$), and remotely significant in one other case ($sigX$). On the contrary, estimations using the BNRC score were strongly significant in 4 cases and remotely significant in the case of $sigF$. This indicates that the capability of the BNRC score to model non-linear interactions is superior to modelling approaches that are restricted to linear interactions.

We note that the estimations for the BNRC score were made without using the time information, while the interval lengths and the time order of arrays were used for the differential equation model. However, if the length of the time intervals chosen in the microarray experiments was not adequate, this might have influenced the estimation results.

2.3.5 Usage of the Prior Probability

We have introduced a general scheme for constructing score functions for Bayesian networks in Equation 2.12 (page 17) and saw that it consists of the marginal likelihood of the data given the model and the prior probability of the model. We saw examples of prior probabilities for the BDe score (Equation 2.16) and the BNRC score (Equation 2.26). In the case of the BNRC score, a high number of parents was penalised in order to avoid overfitting of too extensive networks. This also complies with the biological knowledge that the number of regulators of a gene is usually low. In a similar way, the prior of the BDe score penalises networks with a long description length. However, the description length used here gets low for very small numbers of parents as well as for very high numbers of parents (see Equation 2.13). This can potentially cause the estimation of networks with more parents than biologically plausible.

However, in general the prior probability can be used in a more sophisticated way in order to incorporate more specific previous knowledge. In [Tamada03], information about similar sequences in promoter regions was incorporated in the prior probability. In the same way, it would also be reasonable to make use of existing knowledge about motifs in the promoter region, and knowledge about which proteins can bind the motif. Another example is the work of [Imoto03a, Nariai04], where knowledge about protein-protein interactions was used.

2.4 Estimation of Gene Networks

We first give a general definition of the gene network estimation problem, discuss its computational complexity and the size of its search space in Subsection 2.4.1, and then outline previous work on algorithms for this problem in Subsection 2.4.2.

2.4.1 Gene Network Estimation Problem

After defining the problem in 2.4.1.1, we discuss results on its computational complexity in 2.4.1.2. The size of the search space is discussed in 2.4.1.3.

2.4.1.1 Problem Definition

For any of the three approaches of modelling gene networks, the problem of selecting the most plausible/likely network out of the space of possible networks arises. In order to formally define this optimisation problem, we first need to abstract from the concrete scoring schemes given in Section 2.3, since these scores have different structures and are subject of on-going research. Therefore, an algorithmic approach that depends on the properties of a certain score would have a limited scope.

Definition 14:
Let G be a finite set. We call a function $s : G \times 2^G \to I\!R$ a *score function* for G.
•

Defining the score function in this way for a single gene complies with all scores for Bayesian networks. For the MDL score, we see this directly from Equations 2.13, 2.14, and Equation 2.15. For the BDe score, this follows immediately from Equation 2.19, and for the BNRC score, it is stated in Equation 2.31. However, for Boolean networks and differential equation models, the score can be decomposed to scores for single genes as well. The score for the network is then obtained by adding up the single scores, as we note in the following definition.

Definition 15:
Let s be a score function. Let N be a directed graph. We define the *score of network* N as

$$score(N) = \sum_{g \in G} s(g, Pa^N(g)).$$

•

In order to find the best network, $score(N)$ has to be optimised. For the MDL score and the BNRC score, this was formulated as a minimisation problem, but as a maximisation problem for the BDe score. Here, we define it as a minimisation problem without loss of generality.

Definition 16:
Let G be a set of genes. Let $s : G \times 2^G \to \mathbb{R}$ be a score function. The *gene network estimation problem* is the problem to find a directed acyclic graph N^* that minimises $score(N^*)$. •

In this definition, we restricted to directed acyclic graphs, since we use Bayesian networks for all evaluations of the following chapters. In the case of Boolean networks and dynamic networks, the definition of the gene network estimation problem is analogously for the space of general directed graphs.

2.4.1.2 NP-hardness

The gene network estimation problem is known to be NP-hard, even in the case of the discrete scores MDL and BDe [Chickering96]. Below, we show the NP-hardness of the following problem that evolves in the context of differential equations, when one searches for the best set of matrix entries to be replaced by zero (compare Equation 2.8).

Definition 17:
Let $m, n, k \in \mathbb{N}$, $x^{(j)} \in \mathbb{R}^n$, a $n \times n$ matrix Λ and $t_j \in \mathbb{R}$ for $j = 1, \ldots, m$, where $t_j < t_{j+1}$ holds for all $j = 1, \ldots, m-1$. The *mask selection problem* is the problem to find a $n \times n$ matrix M with entries in $\mathbb{B}$, such that exactly k entries of M equal 1 and

$$\sum_{j=2}^{m} \varepsilon_j^T \cdot \varepsilon_j$$

is minimised, where ε_j denotes

$$\varepsilon_j = x^{(j)} - x^{(j-1)} - (t_j - t_{j-1}) \cdot (M \circ \Lambda) \cdot x^{(j-1)},$$

and $\circ$ denotes the element-wise product. •

By the definition, if the mask selection problem can be solved, Equation 2.8 can be minimised by solving the mask selection problem for all $k = 0, \ldots, n^2$.

Lemma 1

The mask selection problem is NP-hard.

Proof. We reduce the subset sum problem to the mask selection problem. The subset sum problem is the problem to decide for a given set $A = \{a_1, \ldots, a_n\} \subset \mathbb{N}$ and $p \in \mathbb{N}$, whether there is a subset $B \subseteq A$ such that $\sum_{b \in B} b = p$. This problem is known to be NP-hard [Garey79]. Given an input $I = (A, p)$ for the subset sum problem, we define an input for the mask selection problem in the following way. We define an $n \times n$ matrix Λ by

$$\Lambda_{ij} = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{otherwise,} \end{cases}$$

two vectors $x^{(1)}, x^{(2)} \in \mathbb{R}^n$ by

$$x_i^{(2)} = \begin{cases} p + a_1 & \text{if } i = 1, \\ a_i & \text{otherwise,} \end{cases}$$

and $x_i^{(1)} = a_i$ for all $i = 1, \ldots, n$. We also set $t_1 = 0$ and $t_2 = 1$. Given the input $I' = (2, n, k, x^{(1)}, x^{(2)}, \Lambda, t_1, t_2)$ for some $k \in \{0, \ldots, n^2\}$, the mask selection problem is by Definition 17 the problem to minimise $(p - \sum_{i=1}^{n} x_i \cdot a_i)^2$ for $x \in \mathbb{B}^n$. Obviously, there is a subset $B \subseteq A$ such that $\sum_{b \in B} b = p$, if and only if there is a vector $x \in \mathbb{B}^n$ with $(p - \sum_{i=1}^{n} x_i \cdot a_i)^2 = 0$. Therefore, there is a solution for the input I of the subset sum problem if and only if the optimal solution of the mask selection problem for input I' minimises the goal function to zero which shows the claim. △

We conclude that in the case of Bayesian networks there is no polynomial time algorithm that can solve the gene network estimation problem, unless $P = NP$. This also holds in the case of modelling with differential equations, unless it can be shown that the restrictions for Λ and the $x^{(j)}$ given in Equations 2.6 and 2.7 create a feasible special case of the mask selection problem.

2.4.1.3 Search Space Size

As stated in Definition 16, the search space for a gene network of n genes is the space of directed acyclic graphs with n vertices. A recursive formula as well as an

asymptotic expression for the number of directed acyclic graphs with n vertices, denoted as c_n, was derived by Robinson [Robinson73]. We first state the recursive formula:

$$\begin{aligned} c_1 &= 1 \\ c_n &= \sum_{s=1}^{n} (-1)^{s+1} \cdot \binom{n}{s} \cdot 2^{s \cdot (n-s)} \cdot x_{n-s} \end{aligned} \tag{2.37}$$

The asymptotic expression is:

$$c_n = \frac{n! \cdot 2^{\frac{n}{2} \cdot (n-1)}}{r \cdot z^n}; r \approx 0.57436; z \approx 1.4881 \tag{2.38}$$

Some concrete values are shown in Table 2.2. For example, there are roughly $2.34 \cdot 10^{72}$ possible networks with 20 genes, and about $2.71 \cdot 10^{158}$ possible solutions for a gene network with 30 genes. Even for a gene network of 9 genes, there are roughly $1.21 \cdot 10^{15}$ possible solutions. Even if a supercomputer could compute the score for 10^7 network models within one second, searching all models in this space would require 3.8 years of computation time. Therefore, a brute force approach is not feasible, even for a network of 9 genes.

2.4.2 Previous Approaches

Since the gene network estimation problem is NP-hard and the search space of super-exponential size, researchers have so far used heuristic approaches to estimate Bayesian networks [Someren02]. In the special case of dynamic networks, a brute force strategy can be applied for a small number of genes. We first outline heuristic approaches in 2.4.2.1, then discuss the special case of dynamic networks 2.4.2.2.

2.4.2.1 Heuristic Approaches

The most commonly used heuristic approaches in order to estimate gene networks are greedy algorithms [Friedman00, Imoto02, Kim03, Pe'er01, Tamada03], simulated annealing [Hartemink02, Mjolsness00, V.A.Smith02], and genetic algorithms [Someren02], though the latter approach is applied less frequently.

Greedy algorithms start from some initial solution and successively improve it by applying the modification that yields the best improvement of the score function, where the modification is selected from a fixed set of modifications. If none of the modifications under consideration yields an improvement, greedy algorithms terminate and return the last solution.

We state the definition of one algorithm of this kind that has been used in several research projects. The algorithm takes one parameter $n \in \mathbb{N}$.

Algorithm 2: Greedy Hill Climbing [Heckerman95]

Step 1:	Initialise the actual solution as the empty network.
Step 2:	Randomly select a permutation $\pi : \{1, \ldots, \|G\|\} \to G$.
Step 3:	For all $i = 1, \ldots, \|G\|$, do the following two steps:
Step 3a:	Compute the changes of the score when adding a new parent for $\pi(i)$, or removing or reversing the edge of a parent gene of $\pi(i)$.
Step 3b:	Select the modification among the modifications not affecting the acyclicity that improves the score most.
Step 4:	Repeat Step 3 until there score does not improve.
Step 5:	Repeat Step 1 to 4 n times and return the best solution found in these iterations.

Since greedy algorithms may enter local optima easily, adding a random component to the algorithm improves the performance. We note that a greedy algorithm was proposed for the case of dynamic networks as well [deHoon03a].

In contrast to greedy algorithms, simulated annealing also allows modifications that make the score worse. Therefore, such algorithms do not get stuck in local optima fast, but usually require more time for one execution.

However, for none of the algorithms noted above any assertion concerning the ratio of optimal score and the score of the returned solution was made. Therefore, there is no guarantee for high estimation accuracy if such algorithms are applied for the estimation of gene networks.

2.4.2.2 Estimation of Dynamic Networks

In the case of dynamic networks, the search space is the space of directed graphs including cyclic graphs. Since the score function can be written as the sum of the score for each gene and there is no restriction to acyclicity, selection of parents of a gene $g \in G$ can be done without consideration of the parents of other genes in G.[7] Therefore, by computing the local score of each gene g for all $2^{|G|}$ selections of sets of parents from G, optimal dynamic networks can be computed. This was first done in [deHoon03b], though also heuristic algorithms have been applied to the estimation of dynamic networks [deHoon03a, Kim03]. The overall computation time of this approach is in $O(n \cdot 2^n)$.

[7] For gene networks modelled with differential equations, this was shown in [deHoon03b] by modelling the magnitude of error independently for each gene.

2.5 Conclusion

Among the existing modelling approaches, Boolean network models seem to be the most restrictive approach, since the expression data has to be discretised down to only two values. Such discretisation will almost certainly mean loss of information, since actual gene expression levels may show a number of different states. Also, biologically justifiable discretisation thresholds might vary for different genes and discretisation in the presence of high measurement noise leads to a high number of wrongly discretised values. Another problem of Boolean networks is that no mechanism against overfitting is provided.

Modelling gene networks by differential equations allows to treat gene expression data without discretisation and, therefore, avoids loss of information. However, the interaction of genes is modelled in a linear way which contradicts biological knowledge. This is also a likely explanation for the gap of the estimation accuracy observed in Subsection 2.3.4. Furthermore, this modelling approach is restricted to time-course data. Since the space of environmental conditions that can be chosen in experiments has a high number of dimensions (like temperature, concentrations of various molecules, and so on), focusing only on the dimension of time might limit the possibilities of unraveling gene networks. Another problem is that there is no result on the appropriate choice of Δt in time-course experiments.

Modelling gene networks with Bayesian networks allows to treat undiscretised expression data from any kind of experiment. For example, data obtained from gene disruption experiments or gene overexpression experiments can be used, and different kinds of data can be combined as well. It is also possible to model nonlinear gene interactions in this framework, making it the most promising approach towards the elucidation of gene networks.

However, previous algorithmic approaches to the estimation of Bayesian networks were limited to heuristic algorithms. Since heuristic algorithms do not provide any assertions on the quality of the search results, conclusions from heuristic estimations are hard to derive in the presence of this uncertainty.

Number of Vertices	Number of DAGs
1	1
2	3
3	25
4	543
5	29,281
6	3,781,503
7	1,138,779,265
8	783,702,329,343
9	1,213,442,454,842,881
10	$\approx 4.17 \cdot 10^{18}$
11	$\approx 3.15 \cdot 10^{22}$
12	$\approx 5.21 \cdot 10^{26}$
13	$\approx 1.86 \cdot 10^{31}$
14	$\approx 1.43 \cdot 10^{36}$
15	$\approx 2.37 \cdot 10^{41}$
16	$\approx 8.37 \cdot 10^{46}$
17	$\approx 6.26 \cdot 10^{52}$
18	$\approx 9.93 \cdot 10^{58}$
19	$\approx 3.32 \cdot 10^{65}$
20	$\approx 2.34 \cdot 10^{72}$
21	$\approx 3.46 \cdot 10^{79}$
22	$\approx 1.07 \cdot 10^{87}$
23	$\approx 6.97 \cdot 10^{94}$
24	$\approx 9.43 \cdot 10^{102}$
25	$\approx 2.65 \cdot 10^{111}$
$\vdots$	$\vdots$
30	$\approx 2.71 \cdot 10^{158}$
$\vdots$	$\vdots$
40	$\approx 1.12 \cdot 10^{276}$

Table 2.2: The number of directed acyclic graphs.

Chapter 3

Exact Algorithm for Small Gene Networks

In this chapter, we analyse the search space of the gene network estimation problem and develop an algorithm that can compute optimal network models for gene networks of considerable size. Since this result is based on the general definition of the gene network estimation problem introduced in Chapter 2 (Definition 16), it can be applied to any kind of gene expression data using any of the existing statistical modelling approaches. The possibility to compute optimal models opens up the way to address several important research issues as, for example, the question of the best experimental strategy to reveal gene networks.

This chapter is organised as follows. We first introduce and analyse the algorithm in Section 3.1. Then, we discuss issues of the implementation of the algorithm in Section 3.2. Finally, we focus on the algorithm's merits, its computational possibilities and limitations in Section 3.3, and also discuss some results of applications to gene expression data.

3.1 The Algorithm

The basic strategy of the algorithm introduced in this section is to decompose the search space into subspaces and to apply *dynamic programming* to the problem of finding the right subspace as well as to the problem of determining the best solution within a subspace. Dynamic programming is one of the most frequently used algorithmic approaches in computer science and bioinformatics, and has allowed for the solutions to a number of important problems. Some well-known examples from bioinformatics are the Viterbi algorithm for determining a most likely state sequence for hidden Markov models [Viterbi67], the Nussinov algorithm for RNA secondary structure prediction [Nussinov78], and the Smith-Waterman al-

gorithm for local sequence alignment [T.F.Smith81]. Dynamic programming was also applied to classical problems of computer science [Bellman62].

We introduce the decomposition of the search space in Subsection 3.1.1, define the algorithm in Subsection 3.1.2, and analyse the algorithm in Subsection 3.1.3.

Throughout this section, we assume we are given a set of genes G and a score function $s : G \times 2^G \to I\!R$ that induces a score function for networks as defined in Definition 15.

3.1.1 Dividing and Conquering the Search Space

By Definition 16, the task of estimating a network is to find a set of parent genes for each gene, such that the resulting network is acyclic and the score of the network is minimal. We need the following definition.

Definition 18:
We define $F : G \times 2^G \to I\!R$ as

$$F(g, A) = \min_{B \subseteq A} s(g, B)$$

for all $g \in G$ and $A \subseteq G$. •

The meaning of $F(g, A)$ is, by the definition, the optimal choice of parents for gene g, if parents have to be selected from the subset A.

For every acyclic graph, there is a topological sort, i.e. an ordering of the vertices, such that all edges are oriented in the direction of the ordering. Conversely, when given a fixed order of G, we can think of the set of all graphs that comply with the given order, as we do in the next definition. An ordering of a set $A \subseteq G$ can be described as a permutation $\pi : \{1, \ldots, |A|\} \to A$. Let us use Π^A to denote the set of all permutations of A.

Definition 19:
Let $A \subseteq G$ and $\pi \in \Pi^A$. Let $N = (A, E)$ be a network on A. We say N is π-*linear*, if and only if $\pi^{-1}(g) < \pi^{-1}(h)$ holds for all $(g, h) \in E$. •

The basic strategy of our algorithm is to divide the space of all directed acyclic graphs on a set $A \subseteq G$ in subspaces of π-linear networks, for all $\pi \in \Pi^A$. The gene network estimation problem can then be decomposed into the following two problems:

1. Find the subspace of the search space that contains the optimal network searched for. (Lemma 3)

2. Find the optimal network within the selected subspace. (Lemma 2 and Lemma 3)

We will use π to denote a subspace of the search space.

Now we use the above definitions and define function Q^A which will allow us to compute the score of the best π-linear network for a given π, as we show below.

Definition 20:
Let $A \subseteq G$. We define $Q^A : \Pi^A \to I\!R$ as

$$Q^A(\pi) = \sum_{g \in A} F(g, \{h \in A | \pi^{-1}(h) < \pi^{-1}(g)\}).$$

for all $\pi \in \Pi^A$. •

If we can compute the best π-linear network for any given permutation π using functions F and Q, then what remains to do in order to find the optimal network is to find the optimal permutation π which yields the global minimum. Formally, we define function M for this step.

Definition 21:
We define $M : 2^G \to \bigcup_{A \subseteq G} \Pi^A$ as

$$M(A) = \arg \min_{\pi \in \Pi^A} Q^A(\pi)$$

for all $A \subseteq G$. •

Dividing the search space in subspaces might also be used for applying branch-and-bound to the gene network estimation problem. However, one would have to find a decomposition of the search space that allows for the computation of strong lower bounds of the optimal score within a subspace in order to prune significant parts of the search space.

3.1.2 Definition of the Algorithm

Using above notations, the algorithm can be defined as follows.

Algorithm 3: [Ott04a]

Step 1: Compute $F(g, \emptyset) = s(g, \emptyset)$ for all $g \in G$.
Step 2: For all $g \in G$ and all $A \subseteq G - \{g\}$, $A \neq \emptyset$, compute $F(g, A)$ as $\min\{s(g, A), \min_{a \in A} F(g, A - \{a\})\}$.
Step 3: Set $M(\emptyset) = \emptyset$.
Step 4: For all $A \subseteq G$, $A \neq \emptyset$, do the following two steps:
Step 4a: Compute $g^* = \arg\min_{g \in A}(F(g, A - \{g\}) + Q^{A-\{g\}}(M(A - \{g\})))$.
Step 4b: For all $1 \leq i < |A|$, set $M(A)(i) = M(A - \{g^*\})(i)$, and $M(A)(|A|) = g^*$.
Step 5: Return $Q^G(M(G))$.

In the recursive formulas given in Step 2 and in Step 4, we want to compute the function F respectively M for a subset $A \subseteq G$ of cardinality $m = |A|$, and need function values of function F respectively M for subsets of cardinality $m - 1$. Therefore, we can apply dynamic programming in Step 2 as well as in Step 4 to compute functions F respectively M for subsets A of increasing cardinality. In the recursive formula in Step 4, first the last element g^* of the permutation $M(A)$ is computed in Step 4a, and then $M(A)$ is set in Step 4b.

In the computation of function M in Step 4, an order of elements is assembled by consecutively adding a new last element. In an application of dynamic programming to the travelling salesman problem, the computation was done in a similar way for computing shortest paths that pass each vertex in a given set of vertices exactly once [Bellman62]. However, in the case of the travelling salesman problem the problem can be solved in exponential time by a single application of dynamic programming.

Since the computation of F respectively M for subsets of size m only depends on function values for sets of lower cardinality, the computation for all subsets of size m can be done in parallel and in an arbitrary order. This allows to parallelise the computation.

3.1.3 Correctness and Complexity

First, we prove the correctness of the algorithm. The correctness of the recursive formula in Step 2 of the algorithm follows directly from the definition of F. Therefore, after the execution of Step 1 and Step 2, the values of function F for all genes g and all subsets $A \subseteq G$ are stored in the memory. Before proceeding to Step 3 and Step 4, we prove a lemma on the meaning of function Q^A.

Lemma 2 [Ott04a]

Let $A \subseteq G$ and $\pi \in \Pi^A$. Let N^* be a π-linear network on A with minimal score. Then, $Q^A(\pi) = score(N^*)$ holds.

Proof. In a π-linear graph, a gene g can only have parents h which are upstream in the order coded by π, that is, $\pi^{-1}(h) < \pi^{-1}(g)$. Therefore, when selecting parents for g, we are restricted to $B = \{h \in A | \pi^{-1}(h) < \pi^{-1}(g)\}$, and $F(g, B)$ is the score of an optimal choice of parents in this case. Since in a π-linear graph, all edges comply with the order coded by π, we can choose parents in this way for all genes independently which proves the claim. $\triangle$

Using Lemma 2, we now prove that function M can be computed by the formula given in Step 4.

Lemma 3 [Ott04a]

Let $A \subseteq G$. Let $g^* = \arg\min_{g \in A}(F(g, A - \{g\}) + Q^{A-\{g\}}(M(A - \{g\})))$. Define $\pi \in \Pi^A$ by $\pi(i) = M(A - \{g^*\})(i)$, and $\pi(|A|) = g^*$. Then, $\pi = M(A)$.

Proof. Let $\pi' \in \Pi^A$. By the definition of M, we have to show $Q^A(\pi) \leq Q^A(\pi')$. Let N^* be an optimal π-linear network, M^* be an optimal π'-linear network. Then, by Lemma 2, $Q^A(\pi) \leq Q^A(\pi')$ is equivalent to $score(N^*) \leq score(M^*)$. Let us denote the last element of π' as $h = \pi'(|A|)$. We note that for any $B \subseteq G$, $Q^B(M(B))$ is the score of a global optimal network on B by above definitions and Lemma 2. Therefore, we have:

$$\begin{aligned} score(M^*) &= s(h, Pa^{M^*}(h)) + \textstyle\sum_{g \in A - \{h\}} s(g, Pa^{M^*}(g)) \\ &\geq s(h, Pa^{M^*}(h)) + Q^{A-\{h\}}(M(A - \{h\})) \\ &\geq F(h, A - \{h\}) + Q^{A-\{h\}}(M(A - \{h\})) \\ &\geq \min_{h \in A}(F(h, A - \{h\}) + Q^{A-\{h\}}(M(A - \{h\}))) \\ &= F(g^*, A - \{g^*\}) + Q^{A-\{g^*\}}(M(A - \{g^*\})) \\ &= score(N^*), \end{aligned}$$

which shows the claim. $\triangle$

Since Q can be directly computed using F, the algorithm can compute $Q^G(M(G))$ in Step 5. Finally, $Q^G(M(G))$ is the score of an optimal Bayesian network by the definitions and Lemma 2 which shows the correctness.

If the information of the best parents is stored together with $F(g, A)$ for every gene g and every subset $A \subseteq G$, the optimal network can be constructed during the computation of $Q^G(M(G))$. A more efficient way to use the memory will be discussed in Section 3.2.

Theorem 1 [Ott04a]

Optimal networks can be found using $(\frac{|G|}{2}+1) \cdot 2^{|G|}$ dynamic programming steps.

Proof. Let us use n to denote $|G|$. The dynamic programming in Step 1 and Step 2 requires 2^{n-1} steps for each gene, since a gene may not be one of its parents. In each step one score is computed. In the dynamic programming in Step 3 and Step 4 a total number of 2^n steps is needed, where each step involves looking up some previously stored scores. Note that the function Q^A does not need to be actually computed in Step 4a, because $Q^{A-\{g\}}(M(A-\{g\}))$ can be stored together with $M(A-\{g\})$ in previous steps. Therefore, the overall number of dynamic programming steps is $(\frac{n}{2}+1)\cdot 2^n$ for the computation of an optimal network. △

As we see from the definition of Algorithm 3, the time required to perform one dynamic programming step is in $O(n)$, if the time required for the evaluation of the score function s is ignored. Since the exponential number of dynamic programming steps induces a limit for n in practical applications, the time required for one step can be regarded as constant. The space complexity of Algorithm 3 will be addressed in Section 3.2.

We observe that the overall time complexity equals the time complexity of the brute force strategy for the special case of dynamic networks (see 2.4.2.2). Therefore, the result in Theorem 1 equalises the computation time needed in order to find optimal models for general Bayesian networks and dynamic networks.

In biological reality, while the number of children of a regulatory gene may be very high, the number of parents can be assumed to be limited. When we limit the number of parents, the number of score calculations reduces substantially, allowing for the computation of larger networks.

Therefore, we state the following trivial corollary which is practically very meaningful (see also Subsection 3.3.4).

Corollary 1 [Ott04a]

> *Let $c \in \mathbb{N}$ be a constant. Optimal networks, in which no gene has more than c parents, can be found in $(\frac{|G|}{2}+1)\cdot 2^{|G|}$ dynamic programming steps, which require only $O(|G|^{c+1})$ score computations.*

A too high number of parents would in most cases mean overfitting of the model to given data. Since in most score functions overfitting is avoided as discussed in Chapter 2, the assumption of a limited number of parents also has a statistical justification. For example, for the network estimation discussed in Subsection 6.4.6 (page 111) the number of parents was assumed to be at most 5. Among 43 genes only 3 were estimated to have 4 parents, but none was estimated to have 5 parents. Therefore, it is unlikely that the estimation result was affected by restricting the number of parents.

If we do not want to limit the number of parents by a constant, but instead select for each gene a fixed number of candidate parents, the complexity changes as follows.

Corollary 2 [Ott04a]

Let $c \in \mathbb{N}$ be a constant. For each $g \in G$, let $C_g \subseteq G$ be a set with $|C_g| \leq c$. Optimal networks, in which each gene g has parents only in C_g, can be found in $O(2^{|G|})$ dynamic programming steps.

Proof. Since the parents of each gene are selected from a set of constant size, the complexity of the dynamic programming in Step 1 and Step 2 becomes constant. Therefore, the number of dynamic programming steps is in $O(2^{|G|})$. △

Corollaries 1 and 2 are compared in Subsection 3.3.4 in the context of computational possibilities of Algorithm 3.

3.2 Issues of the Implementation

While the analysis of Subsection 3.1.3 is focused on the time complexity of Algorithm 3, consideration of the space requirements is essential in order to maximise the number of genes for which optimal network models can be computed in practical applications. In this section, we address several issues related to memory requirements of implementations of Algorithm 3.

First, we note that the two applications of dynamic programming in Algorithm 3 can be executed in an alternating fashion, since when we compute function M for a set of size m, we only need function values of functions M and F for a set of size $m-1$. Also, the computation of function F for a set of size m only requires the knowledge of function values of F for sets of size $m-1$. Therefore, the computation of both functions can be organised in the following way. We first compute F and M for the empty set. Then, denoting the set of all subsets of size j as layer j, we iteratively compute function M for layer j using previously computed values of F and M for layer $j-1$, and function F for layer j from values of F for layer $j-1$. After the computation of M for layer j, we do not need to store layer $j-1$ of M anymore, and we can also free the memory required for layer $j-1$ of F after the computation of its layer j. Organising the computation in this way, we only need to keep memory for three layers at a time.[1] This reduces the required memory substantially, since storing all layers for both functions at the same time would require space in $O(|G| \cdot 2^{|G|})$.

[1] Here, we regard values of F for all $A \subseteq G$, $|A| = j$, and all $g \in G$ as one layer.

However, storing only three layers at a time forces us to store not only permutations as the function values of M, but the information of entire networks, since deriving the network from the optimal permutation of G in Step 5 becomes impossible without the information of function values of F. Therefore, it is important to develop a space-efficient way to store networks which also allows fast processing of the stored information. In one of the implementations developed for this thesis, we made use of the restrictions of π-linear networks in order to achieve this. Given a subset $A \subseteq G$, a permutation $\pi \in \Pi^A$, and a π-linear network $N = (A, E)$, we store π and E as the concatenation of bitfields $b_1 + \cdots + b_{|A|}$, where b_i corresponds to the gene $\pi(i) \in A$ and is defined as follows. Denoting the number of bits needed to code a gene in G as $l = \lceil \log_2 |G| \rceil$, $b_i \in \mathbb{B}^{l+i-1}$ codes the gene $\pi(i)$ in the first l bits, and the subset of $\{\pi(1), \ldots, \pi(i-1)\}$ that is the set of parents of $\pi(i)$ in N. In this way, π and N can be stored using $|A| \cdot (l + \frac{|A|-1}{2})$ bits, and this coding can be decoded rapidly.

The number of subsets within a layer is highest for $|A| \in \{\lceil \frac{|G|}{2} \rceil, \lfloor \frac{|G|}{2} \rfloor\}$, and storing networks in the described way, therefore, allows to store one solution using

$$\begin{aligned} \frac{|G|}{2} \cdot (l + \frac{\frac{|G|}{2} - 1}{2}) &\leq l \cdot \frac{|G|}{2} + \frac{|G|^2}{8} \\ &= \lceil \log_2 |G| \rceil \cdot \frac{|G|}{2} + \frac{|G|^2}{8} \end{aligned} \tag{3.1}$$

bits in the biggest layers.[2] For higher layers, the space required to store one solution increases, but this is more than equalised by the rapid decrease of the number of subsets within a layer.

Even after employing above strategies, the memory requirement still becomes the bottleneck of computations on available supercomputers, if one does not make use of hard disk space. However, using hard disk space in a naive way would be prohibitive, since the memory access time would decrease substantially, slowing down the computation exceedingly. Therefore, we have made use of hard disk space in some of the computations of this thesis in the following way. We divide layer j in parts consisting of all sets which induce the same subset for a previously selected set $K \subseteq G$ ($|K| \leq j$), i.e. two sets $A_1, A_2 \subseteq G$ belong to the same part of layer j, if and only if $A_1 \cap K = A_2 \cap K$. This yields a division of each layer in $2^{|K|}$ parts, each part corresponding to a subset of K which we denote as the part's key. Having divided layers in this way, we can divide the computation of layer j in separate computations of its parts, and only need to keep parts of layer $j-1$ that match the part's key $K' \subseteq K$ or that match a subset of K' of size $|K'| - 1$, since in a dynamic programming step of Algorithm 3 we only need to look up function values for subsets that differ in exactly one element. Organising

[2] For simplicity, the formula is only given for the case of even $|G|$.

the computation in this way allows efficient use of hard disk space and to execute computations at the limit induced by the time requirement of the algorithm.

We note that we use sets K of different sizes corresponding to the number of subsets within a layer. This avoids the division of small layers in too many parts.

We have implemented Algorithm 3 as a C++ program. As scoring functions, existing implementations of the BNRC score, the BDe score and the MDL score are used.

3.3 Applications and Results

Theorem 1, Corollary 1 and Corollary 2 show the possibility to compute optimal gene network models for gene networks of limited size. This is useful in itself and can be done for any kind of gene expression data and any of the existing score functions.

In order to reveal gene networks in future research projects, it will be important to closely analyse the best way of the statistical modelling (i.e. definition of the score function) and the effectivity of different experimental strategies, such as gene disruption, gene overexpression, various shock conditions, hormone treatment, drug treatment or time-course experiments. Also, combining data of different experiments might increase the effectivity [Miyano03]. If heuristic algorithms are applied to different kinds of data respectively using different score functions, differences in estimation accuracy can not be unambiguously accounted for the differences in the data respectively the modelling approach, since the accuracy of heuristic algorithms depends on the properties of the score function $s : G \times 2^G \to I\!R$, which itself depends on both the data and the modelling approach. On the contrary, computation of optimal models allows to derive certain conclusions from the observed differences and, therefore, allows for the analysis of above problems. Exemplifying such analyses, we compare the significance of estimations using different score functions in this section, compare the significance of estimations based on different kinds of expression data in 5.2.1.3, determine optimal parameters for gene network estimation in 5.2.1.5, and compare gene networks of related species in Subsection 5.2.2.

Though heuristic estimations may in some situations yield optimal solutions depending on the properties of the score function, the number of genes, coincidence, and other factors, we emphasise that even in such cases the uncertainty about the accuracy of estimation remains (since one does not know that the solution is optimal), which hinders conclusions from the estimation result.

This section is organised as follows. We first apply Algorithm 3 to yeast data in Subsection 3.3.1 and Subsection 3.3.2. Comparison of different score functions is exemplified in Subsection 3.3.3. Finally, we discuss the computational possibilities

and limitations of Algorithm 3 in Subsection 3.3.4.

3.3.1 Application to Heat Shock Data

We applied Algorithm 3 to data selected from a data set of 173 microarrays, measuring the response of *Saccharomyces cerevisiae* to various stress conditions [Gasch00]. We selected 15 microarrays from $25°C$ to $37°C$ heat shock experiments and 5 microarrays from heat shock experiments from various temperatures to $37°C$. Then we selected a set of 9 genes, which are involved or putatively involved in the heat shock response [Hohmann03]. Figure 3.1 shows the optimal network with respect to the BNRC score, Table 3.1 shows the annotation of the selected genes.

Figure 3.1: A gene network estimated by Algorithm 3.

We observe that the transcription factor *mcm1* is estimated to regulate three other genes, while it is not regulated by one of the genes in this set, which is plausible. The second transcription factor in our set of genes, *hsf1*, is estimated to regulate three other heat shock genes. It is also estimated to be regulated by a HSP70-protein (*ssa1*), which was reported before [Shi98]. Another chaperone among these genes, *ssa3*, also seems to play an active role in the heat shock response and interacts with *ssa1* and *hsp104*, coinciding with a report by Glover and Lindquist [Glover98].

Overall, the result is biologically plausible and gives an indication for the active role of the chaperones *ssa1* and *ssa3* during the heat shock response. We conclude that optimally estimated gene networks seem to be meaningful and useful for the elucidation of gene regulation.

Gene	Annotation
hsf1	heat shock transcription factor
ssa1	ER and mitochondrial translocation, cytosolic *hsp70*
ssa3	ER and mitochondrial translocation, cytosolic *hsp70*
hig1	heat shock response, heat-induced protein
hsp104	heat shock response, thermotolerance heat shock protein
mcm1	transcription, multifunctional regulator
hsp82	protein folding, *hsp90* homolog
yro2	unknown, putative heat shock protein
hsp26	diauxic shift, stress-induced protein

Table 3.1: Annotation of the selected genes.

3.3.2 Application to Heat Shock and Osmotic Shock Data

In addition to the 20 microarrays from heat shock experiments used in the previous subsection, we additionally selected 10 microarrays from mild heat shock experiments, 7 microarrays from hyper-osmotic shock experiments, and 6 microarrays from hypo-osmotic shock experiments yielding a set of 30 arrays from heat shock experiments and 13 microarrays from osmotic shock experiments. Since we expect the gene network of both, genes involved in heat shock responses and genes involved in osmotic shock responses, to be expressed in the data to some degree, we selected a set of 33 genes involved in one or both of these responses shown in Table 3.2 [Hohmann03].

Response	Selected Genes
heat shock	*hig1, mcm1, msn2, msn4, hsp26, hsp104, tps1, nth1, ctt1, hsp82, ssa1, ssa3, ubc4, msn1, hsf1, gpd1, ssa4*
osmotic shock	*sln1, ssk1, ssk2, ssk22, ptc1, ptc2, ptp2, ptp3, pbs2, ste11, cdc42, cdc24, rck2, hot1, sko1, smp1*

Table 3.2: Genes selected for the gene network estimation. Some of the genes are known to be involved in both responses, though this is not shown.

We have applied Algorithm 3 and obtained an optimal gene network model with respect to the BNRC score which is shown in Figure 3.2.

We find 74 edges in the estimated network. Table 3.3 shows the number of edges within and between the groups of 17 heat shock genes and 16 osmotic shock genes. The number of edges within the heat shock group is higher than the number of edges within the osmotic shock group. Also, heat shock genes (osmotic shock genes) are parents of 40 (34) edges, and children of 48 (26) edges. Therefore, the heat shock genes have clearly more parents than the osmotic shock genes, but not more children. A plausible explanation for the difference in the number of parents might be that there are about twice as many microarrays from heat shock experiments than from osmotic shock experiments.

	Heat Shock Response	Osmotic Shock Response
Heat Shock Response	26	14
Osmotic Shock Response	22	12

Table 3.3: Number of edges within and between both groups. Rows represent parents, columns children.

Figure 3.2 as well as Table 3.3 show a tight interconnection of both groups, which is consistent with the deep relation of both responses [Hohmann03]. As in Figure 3.1, we observe transcription factors (*hot1* and *msn2*) and proteins of the SSA family (*ssa1* and *ssa4*) among the genes with the highest number of estimated interactions.

3.3.3 Comparing the Potential of Score Functions

As a merit of Algorithm 3, we can compare the estimation accuracy of different statistical modelling approaches (score functions) by comparing optimal estimations with respect to some score functions to biological knowledge. In order to evaluate the appropriateness of the BNRC score, a variant of the BNRC score, the MDL score, and the BDe score, we have applied Algorithm 3 to *Bacillus subtilis* microarray data (the time-course data set introduced in Subsection 4.3.1). We selected 80 groups of 7 genes/operons each, and estimated an optimal gene network model for each group and for each of these four score functions (320 optimal estimations overall). Each set of 7 genes consisted of six sigma factors and one operon known to be regulated by one of these as in Section 2.3.4 (page 27) [Sonenshein01] .

In contrast to Section 2.3.4, we did not restrict the number of parent genes, therefore allowing for general directed acyclic graphs. In each estimated network, we checked for an undirected edge connecting the operon with the correct sigma factor. We also counted the number of estimated edges and calculated the probability of hitting the edge under consideration by randomly guessing that number of undirected edges. Using this probability, we applied Bernoulli's formula (Equation 2.36) to compute the p-values of 80 estimations for each score.

In the case of the MDL score and the BDe score, we need to discretise the expression data prior to gene network estimation. In order to make sure that the selection of the discretisation threshold does not bias the comparison, we first selected a threshold that produces a roughly equal number of up-regulated, down-regulated, and normal values. Then we repeated the estimation for some increased and some decreased thresholds and repeated the estimation for all 80 target networks. The results shown in Table 3.4 are the best results among these estimations.

	BNRC score homoscedastic	BNRC score heteroscedastic	MDL score	BDe score
sigE	0.08 (16 out of 22)	0.06 (17 out of 22)	0.76 (1 out of 22)	0.99 (6 out of 22)
sigD	$8.53 \cdot 10^{-3}$ (14 out of 16)	0.02 (14 out of 16)	$3.09 \cdot 10^{-4}$ (6 out of 16)	0.99 (2 out of 16)
sigW	$8.02 \cdot 10^{-4}$ (19 out of 21)	$1.82 \cdot 10^{-3}$ (19 out of 21)	0.74 (1 out of 21)	0.94 (10 out of 21)
sigH	0.07 (9 out of 11)	$3.04 \cdot 10^{-3}$ (11 out of 11)	1.00 (0 out of 11)	0.58 (7 out of 11)
sigX	0.17 (5 out of 6)	0.21 (5 out of 6)	0.32 (1 out of 6)	0.26 (5 out of 6)
sigF	0.09 (4 out of 4)	0.12 (4 out of 4)	1.00 (0 out of 4)	0.97 (1 out of 4)
average number of edges	11.8	12.4	1.3	12.9

Table 3.4: Comparison of the estimation accuracy of four ways of modelling. Values in brackets give the number of operons that were estimated to interact with the correct sigma factor.

In the case of the BNRC score, we observe significant estimations for two sigma factors (*sigD* and *sigW*) and remotely significant estimations for three sigma factors (*sigE*, *sigH* and *sigF*). For the discrete scores MDL and BDe, we only observe one significant obervation (*sigD* in the case of the MDL score). Since

all estimations are optimal estimations, we can conclude that the BNRC score is more appropriate than the discrete scores. The most likely reason for this obvious difference of estimation accuracy seems to be the loss of information during discretisation.

We also included a heteroscedastic variant of the BNRC score in this evaluation [Imoto03b]. Heteroscedastic refers to allowing different variances of the error for different genes. However, the result does not show a significant difference for both scores. The number of correct edges is slightly higher for the heteroscedastic variant, but this corresponds to a slightly higher number of estimated edges.

3.3.4 Computational Possibilities and Limitations

While even networks of small scale like the network estimated in Section 3.3.1 can not be computed using a brute force approach (Equation 2.38), they can be optimally computed using our implementation of Algorithm 3 on a single Pentium CPU with 1.9 GHz for about 10 minutes. In order to evaluate the practical possibilities of this approach, we selected 20 genes with known active role in gene regulation [Lee02] from the yeast data set and estimated a network with optimal BNRC score using all 173 microarrays without restricting the maximal number of parents. The computation finished within about 50 hours using a Sun Fire 15K supercomputer with 96 CPUs, 900MHz each. As a result of this computational experiment, we conclude that the method is feasible for gene networks of 20 genes, even if no constraints are made and a complex scoring scheme like the BNRC score is used. For the discrete scores BDe and MDL which can be computed much faster even networks of little more than 20 genes can be estimated optimally without constraints.

When the maximal number of parents is limited to about 6 (Corollary 1) or, alternatively, sets of about 20 candidate parents are preselected (Corollary 2), even with the BNRC score gene networks of more than 30 genes can be optimally estimated. However, the method as it is now will not allow to estimate networks of much more than about 35 genes with present supercomputers. The gene network of 33 genes in Subsection 3.3.2 was computed on a supercomputer with the above specifications in about 280 hours using 180 gigabyte of memory and about 500 gigabyte of hard disk space. However, the computation could be done faster, if the supercomputer was used exclusively.

Moreover, we note that it is advisable to group genes with nearly identical pattern of gene expression as discussed in Chapter 6. Gene networks that are constructed from such groups, denoted as *network elements*, can be optimally estimated for a higher number of genes, since each network element represents a number of genes.

While the theoretical time complexity of the approach given in Corollary 2 is below the time complexity of the approach given in Corollary 1, we argue that the latter corollary might be practically more important. First, limiting the number of parents by a constant can be easily done and is biologically justified, while selecting a set of candidate parents for each gene requires a method of gene selection, which can potentially bias the computation result. Second, it has to be considered that each dynamic programming step in the computation of function F requires the computation of one score, while one dynamic programming step for function M only requires looking up some previous results. When the number of parents is limited as in Corollary 1, the required number of score calculations becomes a polynomial, which makes this approach faster in practical applications, though the approach in Corollary 2 has a lower asymptotic number of dynamic programming steps.

Figure 3.2: Estimation result of Algorithm 3 applied to heat shock genes and osmotic stress genes.

Chapter 4

Enumerating Optimal Gene Networks

In Chapter 3, it was proved that it is possible to find optimal gene network models for gene networks of considerable size. However, while an optimal gene network model is the network with the best score, there may still be many different networks that could have yielded the same expression data with equal or approximately equal probability. Therefore, one may not assume that given data contains the complete information about a gene network, but one has to bear in mind that the information that can be derived from a given data set might be very well partial information [Friedman03]. Furthermore, even for a gene network of only 10 genes, there are about $4.17 \cdot 10^{18}$ possible network models (see Table 2.2). Thus, even if an optimal network is found, it will in general contain false edges.

In order to overcome this problem, we have developed an extension of Algorithm 3 (page 40) and proved that optimal and suboptimal gene network models can be enumerated in the order of their score, which corresponds to enumerating the most likely networks for usual score functions (see Chapter 2). If we can find common parts of such networks, the likelihood of the common parts is the sum of the likelihood of the networks containing it. Therefore, common parts of the most likely networks, called *gene network motifs* in this work, are expected to be more reliable than single network estimations. This concept is illustrated in Figure 4.1.

This chapter is focused on the enumeration task. Results concerning the extraction of gene network motifs from lists of networks with (sub)optimal are covered in Chapter 5. We first introduce the algorithm and prove its correctness in Section 4.1. We also develop a technique that allows for the enumeration of tens of thousands of the best network models. In Section 4.2, we give an example

enumeration result. Finally, we use the enumeration algorithm for the enumeration of models with the highest scores for randomly selected target networks in order to evaluate the significance of edge counts (the number of networks in a list that contain a given edge) in Section 4.3.

Figure 4.1: The basic idea of gene network enumeration. Different networks may have yielded the same expression data with equal or approximately equal probability. Big light edges represent the generation of data, big dark edges gene network estimation. If all likely networks can be found, common parts can be detected. In this example, the edges $(2, 1)$, $(2, 3)$, and $(5, 4)$ are included in all likely networks.

4.1 Algorithm for the Enumeration of Optimal Gene Networks

We first define the algorithm in Subsection 4.1.1, then prove its correctness and analyse its complexity in Subsection 4.1.2. In Subsection 4.1.3, we develop a technique that allows to increase the number of enumerated networks substantially compared to the formulation of the algorithm in Subsection 4.1.1.

Throughout this section, we assume networks with equal score to be sorted in some arbitrary way, therefore allowing the notion "the k-th best network".

4.1.1 The Algorithm

We show how Algorithm 3 can be improved in order to allow for the enumeration of optimal and suboptimal networks. Since dynamic programming provides the possibility to compute suboptimal solutions as well, we will have to prove that this can also be done in the case of Algorithm 3 that consists of two applications of dynamic programming, where the second application depends on the first one.

We first define some functions and then show how these functions can be computed for gene networks of considerable size. Again, we assume we are given a set of genes G and a score function $s : G \times 2^G \to \mathbb{R}$. For $m \in \mathbb{N}$, we use $\mathbb{N}_{\leq m}$ to denote $\{1, \ldots, m\}$.

Definition 22:
Let $m \in \mathbb{N}$. We define $F^m : G \times 2^G \times \mathbb{N}_{\leq m} \to 2^G$ inductively.[1] First, for all $g \in G$ and $A \subseteq G$, we define

$$F^m(g, A, 1) = \arg\min_{B \subseteq A} s(g, B).$$

Then, denoting the set of all previous solutions $\{F^m(g, A, p) | p < k\}$ as $J(k)$,

$$F^m(g, A, k) = \arg \min_{\substack{B \subseteq A \\ B \notin J(k)}} s(g, B)$$

for all $1 < k \leq m$. •

Definition 23:
Let $m \in \mathbb{N}$. We define $S^m : G \times 2^G \times \mathbb{N}_{\leq m} \to \mathbb{R}$ as

$$S^m(g, A, k) = s(g, F^m(g, A, k))$$

for all $g \in G$, $A \subseteq G$, and $k \in \mathbb{N}_{\leq m}$. •

By the definitions, $F^m(g, A, k)$ is the k-th best choice of parents for a gene g when the parents have to be selected from A, and $S^m(g, A, k)$ is the score for this choice. When m is clear from the context, we use F and S instead of F^m and S^m, respectively. Note that F^m and S^m are partially defined functions, since m may be larger than the number of subsets of A.

As in Chapter 3, we denote a permutation of a set $A \subseteq G$ as $\pi : \{1, \ldots, |A|\} \to A$ and use Π^A to denote the set of all permutations of A. The basic strategy of the algorithm is similar to Chapter 3 (see page 38). The task of finding the best

[1] We define $\mathbb{N}$ as $\{1, 2, \ldots\}$ in this thesis.

networks is divided into the task of finding subspaces that contain these networks and the task to do a search within the subspaces.

In order to find optimal and suboptimal networks for a given subspace specified by permutation π, we need the following function Q^A. For a given gene g, let us denote the set of all genes that precede g in π as $V(\pi, g) = \{h | \pi^{-1}(h) < \pi^{-1}(g)\}$.

Definition 24:
Let $A \subseteq G$. We define $Q^A : \Pi^A \times I\!N^{|A|} \to 2^{A \times A}$ as

$$Q^A(\pi, v) = \{(h, g) | h \in F(g, V(\pi, g), v_{\pi^{-1}(g)})\}$$

for all $\pi \in \Pi^A$ and $v \in I\!N^{|A|}$. •

In Definition 24, we have used a vector $v \in I\!N^{|A|}$ to determine the rank of the selection of parents for the particular genes. Below it will be shown that $Q^A(\pi, v)$ is the set of edges of an optimal or suboptimal π-linear network on A, its rank depending on v. Next, we define two functions M^m and D^m that specify subspaces, in which (sub-)optimal networks can be found, and the choice of a network from the subspace, respectively.

Definition 25:
Let $m \in I\!N$. We inductively define functions $M^m : 2^G \times I\!N_{\leq m} \to \bigcup_{A \subseteq G} \Pi^A$ and $D^m : 2^G \times I\!N_{\leq m} \to \bigcup_{i=0}^{|G|} I\!N^i$ over their second parameter. Let $A \subseteq G$. First, we define

$$D^m(A, 1) = (1, \ldots, 1) \in I\!N^{|A|}$$

and

$$M^m(A, 1) = \arg \min_{\pi \in \Pi^A} score((A, Q^A(\pi, D^m(A, 1)))).$$

Let $k \in I\!N_{\leq m}$ with $k > 1$ and let N be a network on A with optimal score among networks not in $\{(A, Q^A(M^m(A, p), D^m(A, p))) | p < k\}$. Let $\pi^* \in \Pi^A$ be a permutation such that N is π^*-linear. We define

$$M^m(A, k) = \pi^*.$$

Let $v^* \in I\!N^{|A|}$ such that for every $g \in A$, the set of g's parents, $Pa^N(g)$, equals

$$F^m(g, V(\pi^*, g), v^*_{\pi^{*-1}(g)}).^2$$

We define:

$$D^m(A, k) = v^*$$ •

[2]Since network N is among the best m networks on A, the choice of parents for each gene g must also be among the best m choices.

As F^m and S^m, M^m and D^m are partial functions. In Table 4.1, we summarise above notations.

Function	Functionality	Meaning
F^m	$G \times 2^G \times I\!N_{\leq m} \to 2^G$	$F^m(g, A, k)$ is the k-th best choice of parents for g from A
Q^A	$\Pi^A \times I\!N^{\lvert A\rvert} \to 2^{A \times A}$	$(A, Q^A(\pi, v))$ is a π-linear network
M^m	$2^G \times I\!N_{\leq m} \to \bigcup_{A \subseteq G} \Pi^A$	the k-th best network on A is $M^m(A, k)$-linear
D^m	$2^G \times I\!N_{\leq m} \to \bigcup_{i=0}^{\lvert G\rvert} I\!N^i$	the k-th best network on A is $(A, Q^A(M^m(A, k), D^m(A, k)))$

Table 4.1: Functions used to define Algorithm 4. The meanings follow directly from the definitions and Lemma 4.

Using these notations, the algorithm can be defined as follows, given $m \in I\!N$:

Algorithm 4: [Ott04b]

Step 1: Set $F^m(g, \emptyset, 1) = \emptyset$, $S^m(g, \emptyset, 1) = s(g, \emptyset)$ for all $g \in G$.

Step 2: For all $g \in G$ and all $A \subseteq G - \{g\}$, $A \neq \emptyset$, do the following two steps for all $j \leq m$:

Step 2a: Select $B^* \subseteq A$ from
$\{B \subseteq A | B = A \vee B = F^m(g, A - \{h\}, p), h \in A, p \leq m\} - \{F^m(g, A, p) | p < j\}$
such that $s(g, B^*)$ is minimised.

Step 2b: Set $F^m(g, A, j) = B^*$, $S^m(g, A, j) = s(g, B^*)$.

Step 3: Set $M^m(\emptyset, 1) = \emptyset$ and $D^m(\emptyset, 1) = \emptyset$.

Step 4: For all $A \subseteq G$, $A \neq \emptyset$, do the following three steps for all $j \leq m$:

Step 4a: Choose a triple $(g, p, q) \in A \times I\!N_{\leq m} \times I\!N_{\leq m}$ such that
$score((A - \{g\}, Q^{A-\{g\}}(M^m(A - \{g\}, p), D^m(A - \{g\}, p)))) + S^m(g, A - \{g\}, q)$
is minimised and (g, p, q) induces a network different from $(A, Q^A(M^m(A, r), D^m(A, r)))$ for $r < j$.

Step 4b: Set $M^m(A, j)(i) = M^m(A - \{g\}, p)(i)$ for all $i < |A|$, and $M^m(A, j)(|A|) = g$.

Step 4c: Let v denote $D^m(A - \{g\}, p)$. Set $w \in I\!N^{|A|}$ as $w_i = v_i$ for all $i < |A|$ and $w_{|A|} = q$. Set $D^m(A, j) = w$.

Step 5: Return $Q^G(M^m(G, i), D^m(G, i))$ for all $i \leq m$.

The algorithm computes the functions F^m and S^m in Step 1 and in Step 2 for all $g \in G$, $A \subseteq G$, and $j \leq m$. In order to select B^* in Step 2a only function values of F^m for a set A of lower cardinality or lower j are needed. Therefore, F^m and S^m can be computed by applying dynamic programming.

In Step 3 and Step 4, functions M^m and D^m can be computed similarly using dynamic programming, since for the selection of a triple in Step 4a only function values of M^m and D^m for a set A of lower cardinality or lower j are needed in order to compute $M^m(A, j)$ and $D^m(A, j)$. The meaning of the triple (g, p, q) is to specify a network on A as follows. g is a candidate for the last element in the permutation searched for. When g becomes the last element, the remaining permutation can be chosen from up to m previously computed permutations of $A - \{g\}$. The remaining permutation chosen is specified by p. Then, to form a network in the subspace defined by the resulting permutation, the q-th best selection of parents for g is used, while for the other genes parents are selected as indicated by $D^m(A - \{g\}, p)$.

Since all four functions F^m, S^m, M^m, and D^m are partially defined, not for all $j \leq m$ function values can be calculated for sets A of low cardinality. For simplicity, we did not explicitly mention this in the formulation of the algorithm.

As in the case of Algorithm 3, the two applications of dynamic programming in Steps 1 and 2, respectively Steps 3 and 4 can be organised in an alternating way in order to reduce the required memory substantially (see Section 3.2).

When one layer of F^m, M^m, or D^m is computed, this can be done in an arbitrary order for the sets in the layer, and the computation depends only on values in the lower layer. Therefore, Algorithm 4 is well parallelisable.

4.1.2 Correctness and Complexity

We first prove the correctness of Algorithm 4, then discuss its complexity.

4.1.2.1 Correctness

Let us denote the k-th best network on a set $A \subseteq G$ by $N^*_{A,k}$. We first reformulate two lemmata from Chapter 3. Using the above notations, Lemma 2 can also be stated in the following way.

Lemma 4 [Ott04b]

Let $A \subseteq G$ and $\pi \in \Pi^A$. Let N^* be a π-linear network on A with minimal score. Then, $score((A, Q^A(\pi, (1, \ldots, 1)) = score(N^*)))$ holds.

Lemma 3 can be reformulated as:

Lemma 5 [Ott04b]

Let $A \subseteq G$ and $m \in \mathbb{N}$. Let $g^* = \arg\min_{g \in A}(S^m(g, A - \{g\}, 1) + N^*_{A-\{g\},1})$. Define $\pi \in \Pi^A$ by $\pi(i) = M(A-\{g^*\}, 1)(i)$, and $\pi(|A|) = g^*$. Then, $\pi = M^m(A, 1)$.

Using these lemmata, we can prove the correctness of Algorithm 4. We use n to denote $|G|$ and regard the computation that is executed for one $g \in G$ and one $A \subseteq G$ in Step 2, respectively for one $A \subseteq G$ in Step 4 as one dynamic programming step.

Theorem 2 [Ott04b]

Let $m \in \mathbb{N}$. The best m networks can be found using $(\frac{n}{2}+1)\cdot 2^n$ dynamic programming steps, where the complexity of a dynamic programming step depends on m.

Proof. The output of the algorithm, $Q^G(M^m(G, i), D^m(G, i))$, $i \leq m$, are the best m networks on G by the definitions. We only need to prove that the recursive formulas given in the algorithm are correct. The equations given in Step 1 are correct by the definitions of F^m and S^m. When we select a subset of a set $A \subseteq G$ in Step 2, we have basically two choices: the whole set A or a true subset. In the former case, we can compute the score of the choice directly, in the latter, we can use previously computed values of F^m and S^m which gives the correctness of Step 2.

After the execution of Step 2, we have computed all values of F^m and S^m. Using these values, function Q can be computed directly. Therefore, we only need to compute functions M^m and D^m in order to being able to produce the output in Step 5. The equations in Step 3 are again correct by the definitions. We observe that with Lemma 4 in combination with Definition 25, the following equation follows by induction:

$$N^*_{A,k} = Q^A(M^m(A, k), D^m(A, k)) \tag{4.1}$$

From this equation and Lemma 5 we see that the recursion in Step 4 is correct for $k = 1$ (variable j in Algorithm 4). For $k > 1$, we compute the suboptimal permutation $M^m(A, k)$ and the suboptimal choice of parents $D^m(A, k)$ in the same way, restricting to a network not previously chosen. The number of dynamic programming steps is the same as noted in Theorem 1 for Algorithm 3 which completes the proof. △

4.1.2.2 Complexity

By Theorem 2, the number of dynamic programming steps necessary for enumerating the best m gene network models for a gene network of n genes is $(\frac{n}{2}+1)\cdot 2^n$ and does, therefore, not depend on m. Here, we determine the time complexity of the dynamic programming steps which depends on both, n and m. Since there are two applications of dynamic programming in Algorithm 4, we have to distinguish between the dynamic programming step in Step 2 (computation of F^m and S^m) and the one in Step 4 (computation of M^m and D^m).

Let us first consider Step 2. Since the complexity of score computations depends on the given score function s, we assume that score computations can be done in $O(1)$ in order to analyse only the component of the complexity of Step 2 that is independent of s. We also note that in practical applications no score computation is necessary in most iterations of Step 2, since usually the number of parents can be assumed to be bound by some constant (see discussion of Corollary 1).

Lemma 6

Let $g \in G$ and $A \subseteq G - \{g\}$, $A \neq \emptyset$. In Step 2 of Algorithm 4, the iterations for g, A, and all $j \leq m$ can be done in time $O(n \cdot m)$.

Proof. If subsets $B \subseteq G$ of cardinality $|A| - 1$ are coded as numbers in $\{1, \ldots, \binom{|G|}{|A|-1}\}$, Step 2b can be executed in constant time. Therefore, we only need to prove that one selection of a subset B^* in Step 2a can be done in time $O(n)$. Neglecting the case of $B^* = A$ which can be checked for in constant time, the essential part of Step 2a is to select $h \in A$ and $p \leq m$ minimising $S^m(g, A-\{h\}, p)$. In the case of the first iteration ($j = 1$), it is sufficient to check combinations of h and p with $p = 1$. This can be done in time $O(n)$, since $A \subseteq G$ and $|G| = n$. In the following iterations ($j > 1$), it suffices to check for every $h \in A$, the value of S^m for the lowest p such that $F^m(g, A - \{h\}, p)$ has not been selected before. Thus, if the reading positions are kept for each of the $|A|$ lists $(S^m(g, A-\{h\}, 1), \ldots, S^m(g, A-\{h\}, m))$ ($h \in A$), one execution of Step 2a can be done in time $O(n)$ which gives the proof. △

We note that the complexity given in Lemma 6 is minimal, since $O(n \cdot m)$ is the number of previous solutions which all have to be read in the worst case.[3] The next lemma gives the complexity of Step 4.

[3] The $|A|$ lists described in the proof of Lemma 6 may have elements in common.

Lemma 7

Let $A \subseteq G$ with $A \neq \emptyset$. In Step 4 of Algorithm 4, the iterations for A and all $j \leq m$ can be done in time $O(n^3 \cdot m^3)$.

Proof. We observe that Steps 4b and 4c can be done in time $O(n)$, after a triple $(g, p, q) \in A \times I\!N_{\leq m} \times I\!N_{\leq m}$ is selected in Step 4a. Therefore, the execution of Steps 4b and 4c for all $j \leq m$ only requires $O(n \cdot m)$. A straightforward way to find the best m triples is as follows. We keep a list of at most m triples which we initialise as the empty list. Then, we generate all triples $t = (g, p, q)$ and check for each t, whether the network specified by t is different from all networks specified by elements of our list, and add t to the list, if it specifies a new network with a sufficiently low score. Generating all triples in $A \times I\!N_{\leq m} \times I\!N_{\leq m}$ can be done in time $O(n \cdot m^2)$, checking one triple against all up to m triples stored in the list requires m comparisons of networks. Since comparing two networks of n genes requires $O(n^2)$ computation steps in the worst case, the total computation time becomes $O(n^3 \cdot m^3)$. △

From Lemma 6 and Lemma 7, we see that the dynamic programming in Step 4 is more costly. However, the total number of dynamic programming steps is lower for Step 4 by a factor in $O(n)$.

We note that since networks with different scores are different, most network comparisons can be avoided, softening the factor n^3 in practice. Also, since n has to be a small number in order to allow the exponential number of dynamic programming steps, networks could be stored as n numbers, where each number specifies the parents of one gene as a subset of G. This would allow to compare two networks in $O(n)$ steps. However, we did not make use of this approach in our implementation of Algorithm 4 in order to minimise the memory required to store networks (see Section 3.2).

Since the complexity of dynamic programming steps in Step 4 depends on m^3, it would be prohibitive to do an exponential number in n of dynamic programming steps for a very large number of enumerated networks m. However, using the programming technique described in the following subsection, it is sufficient to compute substantially less than m networks in most dynamic programming steps in practical applications.

4.1.3 Taking Advantage of a Combinatorial Explosion

In the two applications of dynamic programming in Algorithm 4, solutions for all subsets A of a superset (G respectively $G - \{g\}$) are computed. In one case (Step 2), a solution is a set of the up to m best choices of parents from A for a

given gene g, in the other case (Step 4), a solution is a set of the up to m best networks on A. The dynamic programming is organised in layers of subsets of equal size $k = 0, \ldots, q$, where q denotes the size of the superset. Therefore, the number of dynamic programming steps needed to complete the computation of one layer is $\binom{q}{k}$. We see that this number gets very low for k near 0 and for k near q, but large for k near $\frac{q}{2}$. As discussed in the previous subsection, the time required for one dynamic programming step depends on m. In order to allow for a large m without increasing the total computation time excessively, we have devised the following strategy: For some $l \leq \frac{q}{2}$, compute only $m' < m$ solutions for the first $q - l + 1$ layers ($k = 0, \ldots, q - l$), but allow up to m solutions during the computation of the last l layers $q - l + 1$ to q.

Algorithm 4 has been formulated for a fixed m in all dynamic programming steps. However, it can be possible to derive more solutions for sets in layer k than were previously computed for sets in layer $k - 1$ based on the principle we explain in the following. In both applications of dynamic programming, solutions for layer k are derived by dividing the search space in subspaces and selecting previously computed solutions for these subspaces.[4] The best m solutions may all belong to the same subspace, but it is also possible that some of these belong to different subspaces. Now let us denote a search space of solutions for one set of one layer in one of the dynamic programming applications as S, and let us assume we have $S_1, \ldots, S_p \subseteq S$ ($p \in \mathbb{N}$) with $\bigcup_{i=1}^{p} S_i = S$. Furthermore, let us denote the j-th best solution within S_i as s_{ij}. We assume we have the knowledge of s_{ij} for all $i = 1, \ldots, p$ and $j \leq m'$. Denoting our knowledge as $K = \{s_{ij} | i \in \{1, \ldots, p\}, j \in \{1, \ldots, m'\}\}$, let $f : K \to \{1, \ldots, |K|\}$ be a bijection such that $f(s) < f(s')$ implies that the score of s is lower than the score of s',[5] i.e. f sorts the known solutions by increasing score. Now let $i^* \in \{1, \ldots, p\}$ such that $f(s_{i^*m'})$ is minimised. We can show that for every solution $s \in K$ with $f(s) \leq f(s_{i^*m'})$, s is the $f(s)$-th best solution in S in the following way. Obviously, since we know $f(s) - 1$ solutions that are better than s, s can not be among the best $f(s) - 1$ solutions. Now, let $r \in S$ denote the $f(s)$-th best solution. We have to prove $r = s$. $r \notin K$ would imply that there is i' such that $score(s_{i'm'}) < score(r)$, since r must belong to at least one subspace. Since $score(r) \leq score(s)$ and $f(s) \leq f(s_{i^*m'})$, this would contradict the minimality of $f(s_{i^*m'})$. Therefore, we have $r \in K$. Since we know $score(r) \leq score(s)$, $f(r) \leq f(s)$ follows. For the $f(s) - 1$ solutions that are better than r, their membership to K follows in the same way as above. Therefore, $f(r) \geq f(s)$,

[4] In the case of Step 2, subspaces are choices of parents sets not including a certain gene. In the case of Step 4, subspaces are sets of π-linear networks for any π that has a certain gene as the last element. We also note that these subspaces have overlaps.

[5] Without loss of generality, we assume that all solutions in S have different scores which corresponds to sorting solutions of equal score in some arbitrary way.

which gives $r = s$.

We see from this argument that in Algorithm 4 different values of m may be used for different layers without losing the correctness, if only the first $f(s_{i^*m'})$ solutions are chosen in the computation for each set A. In general, $f(s_{i^*m'})$ may equal $\min_{i=1}^{p} m(S_i)$, if $m(S_i)$ denotes the number of solutions known for a subspace S_i (m' in above discussion). This worst case corresponds to situations where all best m' solutions fall in the same subspace. On the contrary, in the best case we could derive $O(k \cdot m')$ solutions from known m' solutions in layer $k-1$ in the case of Step 2, and $O(k \cdot m' \cdot m')$ solutions in the case of Step 4. Therefore, in practical applications the number of derivable solutions may vary from the worst case $\min_{i=1}^{p} m(S_i)$ to a quadratic number compared to the previous layer. A quadratic increase in one step would mean a super-exponential increase of computed solutions with increasing layers. We presumed that in the practical case it is unlikely to encounter one of these two extreme cases and therefore hypothesised that there will be a significant increase of derivable solutions, though not a super-exponential increase.

In order to test our hypothesis, we conducted computational experiments according to the general strategy introduced above using some parameters $m' < m$ and $l \leq \frac{q}{2}$. We observed that the number of solutions increased for both applications of dynamic programming, though the dynamic programming of Step 4 exhibited a stronger increase than the dynamic programming of Step 2. In the case of Step 4, we observed exponentially increasing numbers of solutions in many cases. However, setting m' or l too low yielded no significant increase. The observed difference in the behaviour of Step 2 and Step 4 is plausible considering the different best cases stated above.

Based on these observations, we decided to add the following improvement to Algorithm 4 and we introduced parameters $l, m'_P, m'_N, m \in \mathbb{N}$ with the following meanings. For the first $q-l+1$ layers, m'_P solutions are computed in Step 2, and m'_N solutions are computed in Step 4. For the last l layers $k = q-l+1, \ldots, q$, we allowed $m'_P \cdot c_P^{k-(q-l)}$ solutions for Step 2, where c_P is defined as

$$c_P = (\frac{m}{m'_P})^{\frac{1}{l}}. \tag{4.2}$$

For Step 4, we set the number of allowed solutions accordingly. m denotes the number of best networks one wishes to derive and $m \geq \max\{m'_P, m'_N\}$ holds. Therefore, we allow an exponentially increasing number of stored solutions. This setting balances well with the exponentially decreasing number of subsets in a layer, if l is chosen as $l \leq \frac{q}{2}$.

Employing this strategy, we make use of the intrinsic combinatorial explosiveness of the space of directed acyclic graphs in order to compute a high number of best networks. If the combinatorial properties of the search space can help us

to enumerate all networks with good score within some range of the score, we should be able to capture all information that can be derived from the data.

In order to test this improved version of Algorithm 4, we selected sets of genes G in a random manner (as in Subsection 4.3.2 below) and applied the improved algorithm to estimate a number of best gene network models from microarray data. Table 4.2 shows our parameter selection for three computational experiments.

	l	$\|G\|$	m'_P	m'_N	m	c_P	c_N
Test 1	8	18	100	100	2000	≈ 1.4542	≈ 1.4542
Test 2	8	18	100	100	10000	≈ 1.7783	≈ 1.7783
Test 3	10	20	200	200	20000	≈ 1.5849	≈ 1.5849

Table 4.2: Parameters used in three computational experiments and the implied values for c_P and c_N.

Table 4.3 shows the resulting numbers of solutions during dynamic programming. We observe that the number of networks increases exponentially, while the increase for the number of choices of parents (dynamic programming of Step 2) is clearly slighter. Therefore, it seems to be advisable to set m'_P higher than m'_N to guarantee a sufficient number of solutions for Step 2 which will be used in Step 4 to construct networks. This is also indicated by the lower computation time required for Step 2 (Lemma 6).

Since the gap between the number of solutions computed during most dynamic programming steps and the number of solutions finally yielded is very high, we conclude that the strategy of using the combinatorial explosiveness of the search space works well.

A different approach of assessing the reliability of gene network estimations is the bootstrap method, which has been applied to the gene network estimation problem in combination with heuristic algorithms [Friedman00, Pe'er01]. In the bootstrap method, the network estimation is repeated under random changes to the data in order to assess the robustness of estimation. However, if optimal gene network models are used, our method of enumerating many of the best models is computationally advantageous, since we only need to do the (possibly) time-consuming computation once.

Since the number of networks that have to be stored during these computations can be very large, it is especially important to store the networks with low consumption of memory (see Section 3.2).

[6]The number of possible solutions is not 32, since in these computations the maximal number of parents of a gene was set to 4.

Layer	Test 1		Test 2		Test 3	
	Step 2	Step 4	Step 2	Step 4	Step 2	Step 4
0	1 (100)	1 (100)	1 (100)	1 (100)	1 (200)	1 (200)
1	2 (100)	1 (100)	2 (100)	1 (100)	2 (200)	1 (200)
2	4 (100)	3 (100)	4 (100)	3 (100)	4 (200)	3 (200)
3	8 (100)	25 (100)	8 (100)	25 (100)	8 (200)	25 (200)
4	16 (100)	100 (100)	16 (100)	100 (100)	16 (200)	200 (200)
5	31 [6](100)	100 (100)	31 (100)	100 (100)	31 (200)	200 (200)
6	57 (100)	100 (100)	57 (100)	100 (100)	57 (200)	200 (200)
7	99 (100)	100 (100)	99 (100)	100 (100)	99 (200)	200 (200)
8	100 (100)	100 (100)	100 (100)	100 (100)	163 (200)	200 (200)
9	100 (100)	100 (100)	100 (100)	100 (100)	200 (200)	200 (200)
10	125 (145)	100 (100)	137 (177)	100 (100)	266 (317)	200 (200)
11	143 (211)	109 (145)	183 (316)	177 (177)	346 (502)	316 (317)
12	160 (307)	112 (211)	239 (562)	316 (316)	436 (796)	375 (502)
13	197 (447)	266 (307)	306 (1000)	450 (562)	521 (1262)	796 (796)
14	235 (650)	447 (447)	385 (1778)	544 (1000)	652 (2000)	835 (1262)
15	265 (945)	505 (650)	477 (3162)	1032 (1778)	828 (3170)	1039 (2000)
16	301 (1375)	945 (945)	583 (5623)	2199 (3162)	984 (5024)	1557 (3170)
17	n/a (2000)	1375 (1375)	n/a (10000)	3316 (5623)	1059 (7962)	3188 (5024)
18		2000 (2000)		10000 (10000)	1239 (12619)	5212 (7962)
19					n/a (20000)	5795 (12619)
20						18440 (20000)

Table 4.3: Results of three computational experiments. The table entries show the number of solutions for one arbitrarily selected set in a layer. Values in brackets are the allowed number of solutions. The corresponding parameters are given in Table 4.2.

4.2 Example Enumeration Result

We have applied Algorithm 4 to the same data and the same set of genes as we used in Subsection 3.3.1. Figure 4.2 summarises the result of the enumeration of the best 1000 gene network models. Only edges which appeared in at least 300 networks are shown. We observe that five edges were added: (*mcm1*,*hsp104*), (*mcm1*,*yro2*), (*hsf1*,*hsp104*), (*ssa1*,*yro2*) and (*hig1*,*hsp104*). Three of these edges are plausible from the role of *mcm1* and *hsf1* as transcription factors. The edge from *ssa1* to *yro2* is - as in Chapter 3 - plausible from the known active role of chaperones during the heat shock response [Glover98]. Including the new edges, *mcm1* is estimated to regulate five other genes.

Moreover, edges were at least 300 times reversed in two cases: (*ssa3*,*hsp104*) and (*hsp82*,*hsp104*). This may be an indication that the direction of interaction can not be judged from the data in these cases.

The overall structure of the network is consistent among the best 1000 network models which meets our expectation that data from heat shock experiments should be informative with respect to the gene network of the heat shock response.

Algorithm 4 is also extensively applied in the evaluations in Section 4.3, Chapter 5 and Chapter 6.

Figure 4.2: Enumeration result for the gene network of 9 genes from Chapter 3. Circles indicate edges that were reversed in at least 300 out of the 1000 best networks. For these bidirectional edges, the number written to the edge is the sum of the occurences of the edge in both directions.

4.3 Evaluating the Significance of Gene Network Estimations

While analysing single gene network estimation results as, for example, in the previous section, may indicate plausibility of estimations, it does not provide firm proof that gene network estimations are significant. In order to validate that enumerated optimal and suboptimal gene network models contain valuable information about real gene networks, we have applied Algorithm 4 to DNA microarray data, enumerated network models for a number of target networks, and compared the estimations to available knowledge from two databases.

Considering the close relationship of transcription and translation in bacteria, we expect gene network estimations based on RNA data solely to be more suitable for bacteria than for eukaryotes, in which transcription and translation take place at different places and at a different time. Therefore, we chose *Bacillus subtilis*

and *Escherichia coli* as promising targets, for which microarray data as well as knowledge about the gene networks is available.

The evaluations in this section are based on counting the number of occurences of particular edges in a list of (sub)optimal networks. Evaluations which also take the structure of networks into account are pursued in Chapter 5. After introducing the microarray data and the knowledge data used for our evaluations in Subsection 4.3.1, we specify a procedure for the random selection of target networks in Subsection 4.3.2. The evaluation results are presented and discussed in Subsection 4.3.3.

4.3.1 Data

For *E. coli*, we selected the data sets GDS95–GDS100 from the Gene Expression Omnibus [Omnibus]. Changes in gene expression levels were elicited by perturbations of tryptophan metabolism, UV exposure, and novobiocin treatment as shown in Table 4.4 [Courcelle01, Khodursky00a, Khodursky00b]. The overall number of microarrays is 53. We then received data concerning known transcriptional regulation in *E. coli* from the RegulonDB [Salgado01] for comparison with estimation results.

Data Set	Experiment	Number of Arrays
GDS95	perturbation of tryptophan metabolism using strains with mutations that affect tryptophan metabolism	9
GDS96		9
GDS97	experiment designed for the analysis of replication fork movement	3
GDS98	novobiocin treatments	17
GDS99	UV exposure	7
GDS100	UV exposure	8

Table 4.4: Microarray data for *Escherichia coli*.

For *B. subtilis*, we used two microarray data sets. One data set was obtained from time-course experiments under various treatments (see Table 4.5) and includes 70 microarrays.

Experiment	Number of Arrays
cold shock	6
competence	7
glucose and glutamine added during sporulation	6
glucose limitation	5
increased aminoacid availability	8
phosphate and glucose starvation	8
phosphate starvation	6
salt shock	5
sporulation	19

Table 4.5: Microarray data from time-course experiments for *Bacillus subtilis*.

The second data set consists of 99 microarrays from gene disruptant experiments of the following genes:

abh, abrB, acoR, ahrC, alsR, ansR, araR, azlB, ccpA(V), ccpB(S),ccpC, citR, citT, codY, comA, comK, cspB, ctsR, dctR, degU(CM), degU(DSM), deoR, exuR, fur, gerE, glcR, glcT, glnR, gntR, gutR, hpr, hrcA, hutP, iolR, kipR, lacR, levR, lexA, lmrA, lrpA, lrpC, mntR, mtrB, paiA, paiB, perR, phoP, purR, pyrR, rocR, sacT, senS, sigB, sigD, sigE, sigF, sigF2, sigG, sigH, sigI, sigK, sigL, sigM, sigV, sigW, sigW2, sigX, sigY, sigZ, sinR, soj, splA, spo0A, spo0J, spoIIID, spoVT, tenA, tnrA, treR, veg-V, vegT7, xylR, ybbH, ybfA, yesS, yhjM, yotL, ytzE, yufL, yugG, yurK, yvkB, yvrH, ywaE, yyaA, yybA, yybE, yydK, zur

Both data sets are not yet publicly available, though for the time-course data set it is was confirmed that biologically meaningful estimations can be done using this data set [deHoon03b]. We then received a data set of known transcriptional regulation from DBTBS [Ishii01, Makita03].

4.3.2 Selection of Target Networks

From the data set of known regulatory relations for *E. coli*, we selected all relations for which experimental evidence is provided. This yielded a set of 899 known relations. From the *B. subtilis* data set, we selected 840 regulatory relations with evidence in the literature. In order to select parts of these large networks, we applied a random procedure as described in the following.

Since we need to select genes in a way that there are some known regulatory relations connecting the selected genes, we select the first few genes randomly, but then select genes that are connected to the previously selected genes iteratively, expanding the selected partial network in each step by one gene and at least one edge. In each iteration, we select a connected component of the intermediate

network with equal probability, and then select a gene with a known relation to at least one gene in the component randomly, if such a gene exists. Since we should avoid trivial choices of target networks, we choose a gene not connected to the previously selected genes, after five connected genes have been selected in a row.

The selection procedure yields a partially known gene network N of non-trivial structure. We represent N as a matrix. Each pair of genes with a known relationship is represented with 1 in the corresponding entry of the matrix. Pairs of genes with no knowledge are represented with 0 in the entries, but 0.5 is set for the entries for pairs (g, h), for which one or more of the following four conditions hold:

1. g is regulated by h.
2. There is a gene i in the target network that regulates g and h.
3. There is a gene i in the target network that is regulated by g (h) and regulates h (g).
4. Condition 2 or Condition 3 hold for a gene i outside the target network.

Using these conditions, nearly correct estimations are distinguished from wrong estimations. If edge (g, h) is estimated and (h, g) is a known regulatory relation, then the fact that these two genes interact, was correctly estimated (Condition 1). If two genes have a common regulator (Condition 2), then a similar pattern of gene expression for these genes is likely. Therefore, an estimated edge connecting g and h is not correct, but at least the indirect relationship of g and h was detected. In the same way, indirect regulatory relations (Condition 3), possibly via some gene not included in the target network itself (Condition 4), are also not entirely wrong, if estimated.

4.3.3 Evaluation Results

We chose the BNRC score as the score function for the gene network estimations, because data discretisation is not necessary, non-linear gene interactions can be modeled, and the evaluation in Section 3.3.3 yielded the best results for the BNRC score. We applied the procedure described above to select 30 target networks of 10 genes for *B. subtilis* and used Algorithm 4 to enumerate the best 500 network models for each set of genes G_i. For each G_i ($i = 1, \ldots, 30$) and for each possible edge (g, h) (respectively $\{g, h\}$ in the case of undirected edges) in a network on G_i, we then counted the number of occurences of the edge in the 500 estimated networks for G_i. The result of this computation is summarised in Figure 4.3. The x-axis represents the ratio of edge counts to the number of estimated networks

(500), divided in 10 intervals. For each of the three groups 0, 0.5, and 1 specified above, the y-axis gives the proportion of edges of the group that fall into a particular interval.

(a) (b)

Figure 4.3: Result for *B. subtilis*. (a) time-course data set, (b) disruptant data set.

We observe that most edges have an edge count below 50 or above 450, yielding high proportions for the first and the last interval. In these two intervals, group 0 separates clearly from group 0.5 and group 1, while group 0.5 and group 1 do not show a clear separation. This indicates that it is meaningful to distinguish partially correct estimations (group 0.5) from estimations that can not be justified from current knowledge. While group 0 shows a clearly lower proportion in the interval of high edge counts, comparatively many edges of group 0 fall into the group of lowest edge counts. Therefore, for edges with a high edge count it is more likely that they are true than for edges with a low edge count. We, therefore, conclude that the edge counts are meaningful.

We note that several factors limit the strength of separation of group 0 to the other groups. First, even if we do not have knowledge about a regulatory relation (g, h), it still may exist. Therefore, some correct estimations will be classified as wrong, because of the fragmentary knowledge. Second, target networks are selected randomly from the data without consideration of the microarray data at hand. Thus, it is likely that for some of the selected target networks the microarray data does not contain information, because the target network may not have been activated by the given experiments. Third, true regulatory relations might be estimated in an indirect way, for example as transitive edges. In this case, one or more edges in the 0.5 group achieve a high edge count, but some edges in the 1 group do not, weakening the significance of the separation to group 0. However, the observed separation in spite of all these limiting factors is an encouraging outcome.

We take a closer look at the first and the last interval in Table 4.6.

		0-10%			90-100%		
		1	0.5	0	1	0.5	0
time-course	undirected	49.8	48.1	59.4	29.3	24.0	15.6
	directed	68.7	66.0	74.6	7.72	7.27	4.47
disruptant	undirected	57.9	60.4	76.3	24.7	18.7	5.92
	directed	73.6	76.6	87.3	10.2	7.07	2.14

Table 4.6: Proportion of edges in the first respectively last interval for *B. subtilis*.

We observe no significant difference between considering directed edges and considering undirected edges. Moreover, the overall pattern observed above seems to be the same for both kinds of microarray experiments, while the disruptant data yields a stronger separation of group 0. It would be tempting to claim that disruptant data is a stronger experimental method for elucidating gene networks. However, we have to take into account the different number of microarrays, which is 70 for the time-course data set, but 99 for the disruptant data set. Therefore, the observed difference might be explained by a higher amount of data, rather than by the experimental method.

We repeated the selection of target networks and the gene network estimation for *E. coli*. The result is summarised in Figure 4.4 and Table 4.7.

Figure 4.4: Result for *E. coli*.

In contrast to the case of *B. subtilis*, the result for *E. coli* does not show a clear separation of group 0. Likely reasons are the lower number of microarrays (53), and the different kind of experiments conducted. The experiments for *E. coli* are not as various as the experiments for *B. subtilis* and designed for the analysis of very specific functional groups of genes (replication fork movement, tryptophan metabolism, and DNA repair). Therefore, selecting target networks in an unrestricted manner from the database knowledge might be immoderate in the case of *E. coli*.

	0-10%			90-100%		
	1	0.5	0	1	0.5	0
undirected	60.0	51.7	63.3	17.8	24.8	13.7
directed	72.0	74.2	79.1	7.11	7.90	5.54

Table 4.7: Proportion of edges in the first respectively last interval for *E. coli*.

As a consequence of these results, we selected the microarray data sets for *B. subtilis* preferentially for the evaluations in the following chapters.

Chapter 5

Extraction of Gene Network Motifs

In Chapter 4, it was shown that the best m network models can be found for gene networks of considerable size and high m. In this chapter, we discuss how to exploit the information contained in such lists. If common parts of the best networks, denoted as *gene network motifs*, can be detected, their likelihood is the sum of the likelihood of the networks containing it. Therefore, gene network motifs can be expected to be more reliable than single network estimations. This idea is also illustrated in Figure 4.1 (page 54).

We define the term gene network motif and related terms in Section 5.1, and discuss algorithmic approaches for the detection of network motifs. We then evaluate the concept of motif extraction in Section 5.2 by comparing extracted gene network motifs to biological knowledge. Furthermore, we apply motif extraction to data of *B. subtilis* and *E. coli* in order to compare the tryptophan gene network of both organisms.

5.1 Problem Definition and Algorithm

After reviewing related approaches in Subsection 5.1.1, we give our definition of gene network motifs in Subsection 5.1.2 and discuss the computational complexity of their detection. We specify our method of motif extraction in Subsection 5.1.3.

5.1.1 Naive Motifs and Model Averaging

We assume we are given a list of networks $N_i = (G, E_i)$ $(i = 1, \ldots, m)$ on a set of genes G. For $g, h \in G$, let us denote the *edge count* of the edge (g, h) by $m_{(g,h)} = |\{N_i | 1 \leq i \leq m, (g, h) \in E_i\}|$. The evaluation in Section 4.3 was based

on $m_{(g,h)}$ without considering the structures of the given networks. This approach motivates the following definition from [Pe'er01].

Definition 26: [Pe'er01]
Let $N_i = (G, E_i)$ be directed graphs $(i = 1, \ldots, m)$. Let $t \in \mathbb{N}$, and $M \subseteq G \times G$. We call M a *naive motif* with respect to the threshold t, if $m_e \geq t$ for $e \in M$ holds. •

In [Pe'er01], naive motifs were extracted from lists of networks estimated using Algorithm 2 on some modified data sets generated by the bootstrap method. Under the assumption that the edge counts of different edges are independent from each other, it was shown that regions of significantly many edges with high edge counts can be found in gene network estimations. However, this assumption does not account for the fact that the edge counts of all edges connected to the same gene g depend on the expression measurements of g. Therefore, the assumption does not hold and reliable conclusions can not be derived from these results.

Naive motifs count edges of different networks equally without consideration of their score. An approach that uses the scores to weight different networks is the model averaging approach which we explain in the following. From the general scoring scheme introduced in Chapter 2 (Equation 2.11 and Equation 2.12), we see that $exp(-score(N_i))$ is proportional to the posterior probability $P(N_i|Data)$. However, since the constant of proportionality is $P(Data)$, the exact posterior probability can not be easily calculated. Instead, we can compute ratios of posterior probabilities using the following equation.

$$\begin{aligned} \frac{P(N_i|Data)}{P(N_j|Data)} &= \frac{P(Data|N_i) \cdot P(N_i)}{P(Data)} \cdot \frac{P(Data)}{P(Data|N_j) \cdot P(N_j)} \\ &= \frac{P(Data|N_i) \cdot P(N_i)}{P(Data|N_j) \cdot P(N_j)} \\ &= \frac{exp(-score(N_i))}{exp(-score(N_j))} \end{aligned} \tag{5.1}$$

Therefore, we can get to know the ratios of the posterior probabilities for given networks, if we can compute $exp(-score(N_i))$. This would cause an underflow in practical situations, but we can compute $exp(-score(N_i) + c)$ instead for some

appropriate $c \in \mathbb{R}$, since

$$\begin{aligned} \frac{exp(-score(N_i))}{exp(-score(N_j))} &= \frac{exp(c) \cdot exp(-score(N_i))}{exp(c) \cdot exp(-score(N_j))} \\ &= \frac{exp(-score(N_i) + c)}{exp(-score(N_j) + c)} \end{aligned} \tag{5.2}$$

holds. Furthermore, defining

$$w^{(i)} = \frac{exp(-score(N_i) + c)}{\sum_{j=1}^{m} exp(-score(N_j) + c)} \tag{5.3}$$

for $i = 1, \ldots, m$ yields an approximation of the posterior probability of N_i since in the case $m = c_{|G|}$

$$\begin{aligned} \frac{exp(-score(N_i) + c)}{\sum_{j=1}^{m} exp(-score(N_j) + c)} &= \frac{exp(-score(N_i))}{\sum_{j=1}^{m} exp(-score(N_j))} \\ &= \frac{P(Data) \cdot P(Data|N_i) \cdot P(N_i)}{P(Data) \cdot \sum_{j=1}^{m} P(Data|N_j) \cdot P(N_j)} \\ &= \frac{P(Data|N_i) \cdot P(N_i)}{P(Data) \cdot \sum_{j=1}^{m} P(N_j|Data)} \\ &= \frac{P(Data|N_i) \cdot P(N_i)}{P(Data)} \\ &= P(N_i|Data) \end{aligned} \tag{5.4}$$

holds, where $c_{|G|}$ denotes the number of directed acyclic graphs as defined on page 33. Model averaging makes use of this rationale:

Definition 27: [Hartemink02]
Let $N_i = (G, E_i)$ be directed graphs $(i = 1, \ldots, m)$. Let $t \in]0, 1]$, and $M \subseteq G \times G$. Using $w^{(i)}$ as defined in Equation 5.3, we say M is a *model averaging motif* with respect to threshold t if for all $e \in M$

$$t \leq \sum_{\substack{i \\ e \in E_i}} w^{(i)}$$

holds. •

Therefore, model averaging motifs are similar to naive motifs, but the networks are weighted by their approximate posterior probability.

5.1.2 Network Motif Extraction Problem

In naive motifs as well as in model averaging motifs, motifs are constructed from edges that appear in many networks respectively in networks with a high cumulative posterior probability. However, even if each single edge has an edge count respectively weight above some threshold, a motif M composed from such edges does not necessarily have to be frequent in the networks at hand. It is even possible that M is not a subgraph of any of the given networks. Therefore, a motif composed from likely edges may be unlikely itself. As a solution to this problem, we define gene network motifs in a way that a high likelihood of the whole structure is assured:[1]

Definition 28: [Ott04b]
Let $N_i = (G, E_i)$ be directed graphs $(i = 1, \ldots, m)$. Let $t \in \mathbb{N}$, and $M \subseteq G \times G$. We call M a *gene network motif* with respect to the threshold t, if there are $1 \leq i_1 < \ldots < i_t \leq m$ such that $M \subseteq E_{i_j}$ holds for all $j = 1, \ldots, t$. •

Therefore, gene network motifs in the sense of Definition 28 are subgraphs of at least t networks. We can also consider the following generalisation which corresponds to the definition of model averaging motifs.

Definition 29:
Let $N_i = (G, E_i)$ be directed graphs and $w^{(i)} \in \mathbb{R}$ $(i = 1, \ldots, m)$. Let $t \in \mathbb{R}$, and $M \subseteq G \times G$. We call M a *weighted gene network motif* with respect to the threshold t, if

$$t \leq \sum_{\substack{i \\ M \subseteq E_i}} w^{(i)} \tag{5.5}$$

holds. •

Definition 28 leads to the following problem which we refer to as the *gene network motif extraction problem*:

> Given graphs $N_1 = (G, E_1), \ldots, N_m = (G, E_m)$, and $k, t \in \mathbb{N}$,
> find a gene network motif M with respect to t such that $|M| = k$,
> if existing.

The motif extraction problem is equivalent to the well-known problem of finding maximal frequent item sets from data mining. The problem of finding balanced

[1] Our definition of gene network motif is, at first, different from the biological definition used, for example, in [Milo02], but might turn out to be closely related.

complete bipartite subgraphs [Garey79] can be reduced to both problems which are, therefore, NP-hard as we show in the following lemma for the motif extraction problem.

Definition 30: [Garey79]
Let $N = (G, E)$ be an undirected, bipartite graph and $K \in \mathbb{N}_{\leq |G|}$. The *balanced complete bipartite subgraph problem* is the problem to decide, whether there are disjoint $G_1, G_2 \subseteq G$ such that $|G_1| = |G_2| = K$ and such that $g \in G_1$, $h \in G_2$ implies $\{g, h\} \in E$. •

Lemma 8

The gene network motif extraction problem is NP-complete.

Proof. Let $N = (G, E)$ be an undirected, bipartite graph and $K \in \mathbb{N}_{\leq |G|}$. Let $H_1, H_2 \subseteq G$ denote the bipartite decomposition of G (see Definition 7), let $h_1^{(1)}, \ldots, h_{|H_1|}^{(1)}$ denote the elements of H_1, and $h_1^{(2)}, \ldots, h_{|H_2|}^{(2)}$ the elements of H_2. Let h^* be an element not contained in H_1 and let G' denote $H_1 \cup \{h^*\}$. We define $m = |H_2|$ directed graphs $N_i = (G', E_i)$ $(i = 1, \ldots, m)$ by

$$E_i = \{(h_j^{(1)}, h^*) | \{h_j^{(1)}, h_i^{(2)}\} \in E\}. \tag{5.6}$$

Let us consider $N_1, \ldots, N_m$ as an input for the gene network motif extraction problem. We show that there exists a motif of size K with respect to threshold K if and only if there exist $G_1, G_2 \subseteq G$ such that $|G_1| = |G_2| = K$ and such that $g \in G_1$, $h \in G_2$ implies $\{g, h\} \in E$. If there is a motif $M = \{(h_{j_1}^{(1)}, h^*), \ldots, (h_{j_K}^{(1)}, h^*)\}$ in K graphs $N_{l_1}, \ldots, N_{l_K}$, then for each $p, q \in 1, \ldots, K$, $\{h_{j_p}^{(1)}, h_{l_q}^{(2)}\} \in E$ holds which shows that $G_1 = \{h_{j_1}^{(1)}, \ldots, h_{j_K}^{(1)}\}$ and $G_2 = \{h_{l_1}^{(2)}, \ldots, h_{l_K}^{(2)}\}$ would form a solution of the balanced complete bipartite subgraph problem. In the same way, the existence of a complete bipartite subgraph of N specified by two sets $G_1, G_2 \subseteq G$ implies the existence of a motif in $N_1, \ldots, N_m$ which completes the proof. △

However, for small gene networks the problem can be solved for many practical instances using the exhaustive search strategy we describe in the following.

5.1.3 Network Motif Extraction Algorithm

In order to extract the partial information about gene networks contained in given gene expression measurements, we propose the following strategy, which

uses three parameters $m, c, k \in \mathbb{N}$.

Algorithm 5: [Ott04b]

Step 1: Enumerate the best gene network models N_i, $1 \leq i \leq m$. (Theorem 2)
Step 2: For every $g, h \in G$, count the occurences of the edge (g, h) in the networks N_i.
Step 3: Select all edges (g, h) with at least c occurences.
Step 4: For all subsets M of the set of selected edges with $|M| = k$, count the networks including all edges in M.
Step 5: Return all motifs M with at least c occurences.

This algorithm makes use of the fact that every edge in a motif M with at least c occurences must have at least c occurences itself. Therefore, the number of candidate edges for composing motifs with more than c occurences can be essentially reduced in practice. The running time of this exhaustive search strategy gets infeasible when the threshold c is set too low, but this did not impose practical limitations for most of our computations, since we are interested in gene network motifs with high confidence.

However, if the number of genes gets too large, Algorithm 5 can not be applied, even for a large threshold c. Since the motif extraction problem is equivalent to the problem of finding maximal frequent item sets, search strategies for this problem can be applied instead [Lin02].

5.2 Network Motif Extraction Results

In this section, we make use of Algorithm 4 to compute lists of best gene network models for a number of target networks, and extract naive motifs, model averaging motifs, and gene network motifs from these lists. In Subsection 5.2.1, we compare extracted motifs to knowledge in order to evaluate their significance and show results of several related evaluations. In Subsection 5.2.2, we extract gene network motifs for a group of genes that is involved in tryptophan metabolism in *B. subtilis* as well as in *E. coli*, and compare the results.

5.2.1 Finding Network Motifs in Bacterial Gene Networks

As in Section 4.3, we focus on bacteria because of the close relationship of transcription and translation in these organisms. In order to compare the significance of extracted motifs to the significance of single optimal gene network models, we employ a general scheme for comparing the strength of different gene network

estimation methods which is introduced in 5.2.1.1. This scheme can also be employed to evaluate changes in significance, when the same estimation method is applied to different data sets.

In the following, we conduct comparisons of the estimation accuracy of motif extraction versus optimal gene network models (5.2.1.2), the significance of estimations from time-course data versus disruptant data (5.2.1.3), and the estimation accuracy for different numbers of microarrays (5.2.1.4). Finally, we assess optimal parameters for motif extraction in 5.2.1.5.

5.2.1.1 Evaluation Scheme

Let us assume we are given two gene network estimation methods A and B, possibly based on different data. Let us also assume we are given some set of available genes $\mathcal{G}$ and knowledge about existing gene interactions $\mathcal{K} \subseteq \mathcal{G} \times \mathcal{G}$. We also require that both estimation methods can do estimations for any subset $G \subseteq \mathcal{G}$.

In order to test both methods, we can employ the following approach. For some $k \in \mathbb{N}$, we select subsets $G_1, \ldots, G_k \subseteq \mathcal{G}$ in the random way described in Subsection 4.3.2 (page 68), and apply both estimation methods to estimate gene network models $N_1^{(A)}, \ldots, N_k^{(A)}$ and $N_1^{(B)}, \ldots, N_k^{(B)}$ for the selected subsets. In the next step, we randomly select one edge $e_i^{(A)}$ from $N_i^{(A)}$ and one edge $e_i^{(B)}$ from $N_i^{(B)}$ for each $i = 1, \ldots, k$. The probability of guessing a single edge of the given knowledge $K_i = (G_i \times G_i) \cap \mathcal{K}$ with respect to the target network on G_i is $p_i = \frac{|K_i|}{|G_i| \cdot (|G_i|-1)}$, since $|G_i| \cdot (|G_i| - 1)$ is the number of possible edges in a directed graph without self-loops. We then count the number of edges $e_i^{(A)}$ that coincide with the knowledge

$$n_A = |\{i | 1 \leq i \leq k, e_i^{(A)} \in K_i\}|, \tag{5.7}$$

and do the same for method B. We compute an upper bound for the probability of guessing at least l single edges correctly among k networks, $P(k, l)$, by using the inequality

$$P(k, l) \leq \sum_{i=l}^{k} \binom{k}{i} p^i (1-p)^{k-i}, \tag{5.8}$$

where p denotes $\max_{i=1}^{k} p_i$. We then compare n_A and n_B as well as the upper bounds for $P(k, n_A)$ and $P(k, n_B)$ and judge one of the methods as stronger if the values deviate significantly and k is chosen large enough in order to minimise bias caused by the random selection of edges from the estimations.

Since we only select one edge from each estimated network, this evaluation scheme can also be applied in situations where the number of edges in estimations

by method A and method B is different. Selecting several edges would also cause difficulties, since several edges from an acyclic graph may not be modelled as independent.[2] We note that results of this evaluation scheme hold for the estimated networks as a whole, since we select arbitrary edges randomly.

If the number of known edges $|K_i|$ in selected target networks varies strongly for different target networks, the upper bounds for $P(k, n_A)$ and $P(k, n_B)$ might get too weak. Therefore, we avoid selecting target networks with an excessive number of known edges in order to allow the computation of tight upper bounds.

5.2.1.2 Significance of Motifs

We have employed above evaluation scheme to compare optimal gene network estimations computed with Algorithm 3 versus motifs extracted from lists of best network models computed with Algorithm 4. Gene network motifs were extracted using Algorithm 5 and we chose motifs with the highest score among the motifs with the most edges. We used the data from DBTBS as knowledge database $\mathcal{K}$ and also accepted indirectly correct edges as specified in Subsection 4.3.2, since the evaluation in Subsection 4.3.3 showed no substantial differences between directly correct edges and indirectly correct edges. We estimated networks from the *B. subtilis* disruptant data set introduced in Subsection 4.3.1.

We selected 1000 target gene networks of 10 genes, enumerated the best 100 networks for each target network, and extracted motifs of 2 or more edges present in at least $t = 80$ networks. We then conducted the same evaluation a second time with $t = 95$ as the threshold for motif extraction. Table 5.1 shows the result.

Method	Number of Selected Edges		Number of Correct Edges		p-value	
	80	95	80	95	80	95
directed edge from directed motif	1000	997	552	581	$2.32 \cdot 10^{-22}$	$2.32 \cdot 10^{-31}$
undirected edge from undirected motif	1000	1000	538	564	$9.05 \cdot 10^{-19}$	$1.04 \cdot 10^{-25}$
directed edge from optimal model[3]	1000	1000	486	459	$2.25 \cdot 10^{-8}$	$8.76 \cdot 10^{-5}$

Table 5.1: Evaluation results for estimations of 1000 target networks of 10 genes using two different thresholds for the motif extraction.

In the case of the increased threshold, no motif scoring above the threshold was found in three cases. Since our random selection of target networks is likely to

[2] In [Rung02], an indirect approach was used to compare an estimated network to knowledge. Random networks with similar properties as the estimated network were generated and the agreement of random networks to knowledge was compared to the agreement of the estimated network to knowledge.

[3] The optimal model does not depend on the threshold of motif extraction, but the evaluation was performed two times.

choose some networks that are not well expressed in the data, we can not expect to find high scoring motifs for any set of genes. Therefore, we must allow gene network estimators to refuse estimation in some cases, since this might be the correct behaviour.

We observe that the number of correct edges is significantly higher than for random estimates. This holds in the case of optimal network models as well as in the case of gene network motifs. However, in the case of motif extractions the number of correct edges is clearly higher than for optimal models, and the p-value is much stronger. We conclude that gene network motifs are even more reliable than the network with best score.

The motifs extracted using the higher threshold $t = 95$ show an increased reliability. The relation of threshold selection and reliability is, therefore, analysed more closely in 5.2.1.5.

In Table 5.1, results for undirected motifs are shown as well. For undirected motifs, directed graphs are transformed to undirected graphs prior to motif extraction. Though in Table 5.1 undirected motifs seemingly perform worse, this observation could not be confirmed in further evaluations.

In order to test whether the relaxed acceptance criteria that allows indirectly correct edges might have influenced above results, we conducted an evaluation under the strict acceptance criteria that judges only direct matches to the database knowledge as correct. For this evaluation, we selected 700 target networks of 10 genes randomly. The motif threshold was set to 90. Table 5.2 shows the result.

Method	Number of Selected Edges	Number of Correct Edges	p-value[4]
directed edge from directed motif	699	99	$2.97 \cdot 10^{-4}$
undirected edge from directed motif	699	213	$3.43 \cdot 10^{-11}$
undirected edge from undirected motif	700	198	$9.54 \cdot 10^{-8}$
directed edge from directed naive motif	700	98	$4.72 \cdot 10^{-4}$
undirected edge from directed naive motif	700	211	$1.22 \cdot 10^{-10}$
undirected edge from undirected naive motif	700	178	$2.84 \cdot 10^{-4}$
directed edge from optimal model	700	74	0.325
undirected edge from optimal model	700	173	$1.35 \cdot 10^{-3}$

Table 5.2: Evaluation results for estimations of 700 target networks of 10 genes applying the strictest acceptance criteria. Since the proportion of known edges is higher in undirected graphs, more undirected edges than directed edges are correct.

Since the strict acceptance criteria based on fragmentary knowledge will reject

[4]In this computation the upper bound given in Equation 5.8 was very rough, since the proportion of known edges p_i deviated substantially for very few networks, yielding a high value for $p = \max_{i=1}^{k} p_i$. In order to get more realistic upper bounds, we replaced p by an upper bound of more than 92% of the p_i.

many correctly estimated edges, and the microarray data does probably contain only few information for many of the target networks, the absolute numbers of correct edges are not high compared to the number of selected edges. However, there are significantly more correct edges than expected under the null hypothesis of meaningless estimators which gives further evidence of the significance of gene network estimations from gene expression measurements. Moreover, as in the case of the relaxed criteria, we observe a substantial difference between optimal models and motif extraction.

As another conclusion from the result in Table 5.2, we see that naive motifs do not perform worse than exact motifs (Definition 28).[5] Therefore, though exact motifs are theoretically superior (as discussed in Subsection 5.1.2), it does not seem to make a significant difference in practice. This is a positive result considering the computational complexity of the extraction problem for gene network motifs.

Table 5.3 shows the result of an evaluation with 200 target networks of 15 genes. In this evaluation, we applied the relaxed acceptance criteria, enumerated the best 200 gene networks and set the threshold for motif extraction to $t = 195$. We observe better results for motif extraction in consistence with above results.

Method	Number of Selected Edges	Number of Correct Edges	p-value
directed edge from directed naive motif	200	107	$7.77 \cdot 10^{-5}$
undirected edge from undirected naive motif	200	103	$6.51 \cdot 10^{-4}$
directed edge from optimal model	200	79	0.584

Table 5.3: Evaluation results for estimations of 200 target networks of 15 genes.

5.2.1.3 Comparing Experimental Techniques

We also employed the evaluation scheme from 5.2.1.1 in order to compare gene network estimations based on time-course data with estimations based on disruptant data. We used the data sets introduced in Subsection 4.3.1, but removed the data of 29 randomly selected microarrays from the disruptant data set in order to equalise the number of arrays to 70 in both data sets. Table 5.4 shows the

[5]The difference in the case of undirected naive motifs was not consistent in further evaluations and can, therefore, be ignored.

result. 100 networks were enumerated for each target network, the threshold for motif extraction was set to 80.

Method	Number of Correct Edges		p-value	
	time-course	disruptant	time-course	disruptant
directed edge from directed motif	435	484	0.013	$4.57 \cdot 10^{-8}$
undirected edge from undirected motif	419	481	0.116	$1.29 \cdot 10^{-7}$
directed edge from optimal model	403	420	0.435	0.104

Table 5.4: Evaluation results for estimations of 1000 target networks of 10 genes estimated from time-course data respectively disruptant data.

We observe a higher number of correct estimations for the disruptant data set, in the case of directed motifs as well as in the case of undirected motifs. Therefore, we conclude that the disruptant data set contains more information with respect to the gene network of the set of 525 genes, for which there are entries in the DBTBS. This result is consistent with the observation in Subsection 4.3.3 that the disruptant data yielded are better discrimination of correct (respectively justifiable) edges from wrong edges.[6] However, it is still possible that the time-course data keeps more information than the disruptant data with respect to some local structures of the whole network.

5.2.1.4 Influence of the Number of Microarrays

Since the evaluation in 5.2.1.2 for $t = 80$ and the evaluation in 5.2.1.3 for disruptant data were performed with equal parameter settings, they only differ in the number of arrays used. Therefore, we can assess the influence of the number of microarrays on estimation accuracy by comparing both results as shown in Table 5.5.

As expected, the estimation accuracy improves with an increasing amount of data. This increase is quite steep, considering the increase of the amount of data of about 41% and suggests further improvement, if the data set is augmented. This result differs from the saturation-like shape observed in Figure 2.2 (page 16), but in the latter evaluation we focused on a single network of six genes instead of many different real networks.

5.2.1.5 Optimal Parameters for Motif Extraction

Since we saw from the above results that extracting motifs from lists of best networks is an effective approach to maximise the agreement of estimations and

[6] In Subsection 4.3.3, we have used a larger set of arrays for the disruptant data.

Method	Number of Correct Edges		p-value	
	70	99	70	99
directed edge from directed motif	484	552	$4.57 \cdot 10^{-8}$	$2.32 \cdot 10^{-22}$
undirected edge from undirected motif	481	538	$1.29 \cdot 10^{-7}$	$9.05 \cdot 10^{-19}$
directed edge from optimal model	420	486	0.104	$2.25 \cdot 10^{-8}$

Table 5.5: Comparison of the accuracy of estimations from a data set of 70 respectively 99 microarrays.

knowledge, we now ask for the optimal parameters for motif extraction. For this purpose, we selected 1000 target networks of 12 genes and estimated these from the disruptant data set for *B. subtilis*. In each estimation, we enumerated the 150 best models and extracted model averaging motifs from these.

If microarray data contains much information about a target network, we expect the best models to be very similar, and, therefore, presume that large motifs can be found. Therefore, we asked, whether the estimation accuracy can be improved, if estimations are refused in cases where a motif of at least l edges can not be found. Figure 5.1 shows the result for varying threshold l. The threshold for motif extraction was set to $t \approx 0.27$.

(a) (b)

Figure 5.1: Results for varying threshold of motif size. (a) number of correct edges, (b) number of selected edges.

We observe that the number of correctly estimated edges decreases with increasing l, but this change is the result of an increasing number of refused estima-

tions, as we also verified by inspecting the p-values for varying threshold l. We conclude that the existence of large motifs does not imply an increased reliability of edges within these motifs.

We also asked for the effect of increasing the threshold for motif extraction and extracted model averaging motifs for varying threshold from 40 (corresponding to the floating point threshold $t = \frac{40}{150} \approx 0.27$) to 150. The result is shown in Figure 5.2.

(a) (b)

Figure 5.2: Estimation accuracy for varying threshold for motif extraction. (a) negatived $\log_{10}$ of p-value, (b) number of correct edges.

We observe no significant trend for low threshold values, while the accuracy steeply increases as the threshold approaches 1.0. Estimation results are in significant agreement to knowledge for all threshold values used, but setting the threshold t to 1.0 or at least close to 1.0 yields the most reliable results.

Finally, we examined the influence of increasing the number of enumerated networks m. We extracted naive motifs and model averaging motifs for all values of m from 1 to 150 using the threshold $t = 0.8$ (corresponding to $t = 120$ in the case of naive motifs). Table 5.6 shows the summarised result as averages from five intervals for m.

For both motif extraction techniques, the estimation accuracy increases with increasing m. Therefore, the best accuracies are achieved for m close to 150. However, since the scope of this computation was limited to $m \leq 150$, further improvement of the accuracy for $m > 150$ can not be excluded.

Interval	Naive Motifs	Model Averaging
1–30	512.57	506.40
31–60	516.83	511.10
61–90	522.13	513.17
91–120	529.03	515.17
121–150	525.20	517.53

Table 5.6: Average number of correct edges, when the number of enumerated best networks varies within an interval.

5.2.2 Comparing Gene Networks of Related Species

In order to analyse gene networks of two related species with respect to similarities and differences, microarray data containing information about the target network is necessary for both species. We searched microarray databases for an appropriate combination of experimental data and decided that the following data is the most promising available combination.

For Gram-negative rod-shaped *E. coli*, there is data from experiments using strains with mutations that affect tryptophan metabolism (introduced in Subsection 4.3.1). The amount of data is low (18 microarrays), but the experiments are very specifically designed to unravel details of the regulation of the genes involved in tryptophan metabolism. Furthermore, we selected 59 microarrays from the time-course data set for Gram-positive rod-shaped *B. subtilis* which include the arrays for all experiments listed in Table 4.5 (page 68) except of the salt shock and the cold shock experiments. The selected microarrays are obtained from bacteria growing on media with various nutritional conditions. Therefore, also differences in the tryptophan levels in the media are to be expected, affecting the tryptophan gene network. The experiments for *B. subtilis* are not as specific as the experiments for *E. coli*, but the number of microarrays is comparatively high.

Since information about the well-studied tryptophan gene network should be deducible from both data sets to some degree, we selected a set of genes known to play a metabolic or regulatory role in tryptophan biosynthesis. We applied Algorithm 5 to both data sets, setting the number of enumerated networks (parameter m) to 100, the threshold for motif extraction (parameter c) to 50, and varied the size of the motif (parameter k) over the range of all sizes, for which motifs above the threshold are found. From the extracted motifs, we selected the highest-scoring motif among the motifs with the highest number of edges. This yielded a motif of 13 edges appearing in 54 of the 100 best networks in the case of *E. coli* (shown in Figure 5.3). The edge counts range from 79 to 100. For

B. subtilis, the motif depicted in Figure 5.4 was found which is a subgraph of 61 of the best networks and consists of 16 edges with counts ranging from 77 to 100.

The genes *trpA*, *trpB*, *trpC*, *trpD*, and *trpE* are members of the same operon in both organisms. Correspondingly, there are many edges connecting these genes in both graphs coinciding partially with the order of these genes within their operons.

Since there are seven genes of *B. subtilis* known to be enzymes involved in the biosynthesis pathway of tryptophan, but only five of these seem to be involved in tryptophan biosynthesis in *E. coli*, it is interesting to take a closer look at the two additional enzymes *trpF* and *pabA*. *tprF* is connected to *trpB* and *trpD*, corresponding to its approximate position in the trp-operon. Though *pabA* is not a member of the trp-operon, but of the folate operon [Gollnick02, Henner93], it is also linked to two genes of the trp-operon (*trpA* and *trpB*). Since this is not the case in the *E. coli* motif, we presume that these edges are in relation to the close functional relation of these genes in the tryptophan pathway of *B. subtilis*, while the missing edges in the case of *E. coli* are consistent with the fact that no involvement of *pabA* has been found.

mtrB is estimated to have a relation to *trpE*, *trpF*, and *dnaJ* in the *B. subtilis* motif.[7] The gene product of *mtrB*, the tryptophan RNA-binding attenuation protein (TRAP), is known as a key regulator of tryptophan biosynthesis [Valbuzzi01]. Since in the motif for *E. coli*, *mtr*, coding a tryptophan specific transport protein, is also connected to *trpE* and *dnaJ*, and has four other connections as well, it seems to have a similar regulatory role in *E. coli* as well.

Recently, a TRAP-inhibitory protein (anti-TRAP, AT) has been found and has been identified as the gene product of *yczA* [Valbuzzi01]. However, *yczA* is not linked to any other gene in the motif for *B. subtilis*. Since the estimations are based on mRNA measurements, the regulatory function of AT might be not observable from the data, if AT acts without a change in its mRNA level. This would be consistent with the protein-binding role of AT that is dependent on tryptophan levels. Similarly, there are more known interactions of *trpR* to other genes in the motif of *E. coli*, but *trpR* is known to act by conformational changes of its protein rather than by changes of its mRNA levels. However, at least the estimated interaction with *trpD* matches knowledge, though the direction of the interaction is known to be opposite.

No *yczA* homologue has been found in *E. coli*, but BLASTP searches have shown that the protein sequence of AT shows similarities to that of cystein-rich domains of the chaperone *DnaJ* [Martinez-Y.00, Szabo96]. *dnaJ* is connected to the motif for both organisms, and it is connected to *mtr* respectively *mtrB*

[7] We note that the edge $(mtrB, trpE)$ matches the only entry for the given genes in DBTBS, though the direction is reversed.

in both cases, though the direction of these edges is opposite. This observation in combination with the similarity to AT on the protein level might indicate a protein interaction of *dnaJ* to *mtr* respectively *mtrB*.

trpS encodes tryptophanyl-tRNA synthetase. tRNA^{Trp} was shown to play an important role in the tryptophan network which differs for the two organisms [Valbuzzi01]. This is consistent with the differential position of *trpS* in both graphs, linking it to *wrbA* and *mtr* in the case of *E. coli*, but to *pabA* and *dnaJ* in the case of *B. subtilis*.

We only observe one connection of *trpL* to another gene (*mtr*) in the motif for *E. coli*. This is interesting, since the tryptophan specific transport protein *mtr* [Ong02] may be involved in transcription termination at the leader peptide (coded by *trpL*) which is observed in abundance of tRNA^{Trp} [Valbuzzi01].

Concluding this analysis, we see that comparing estimated gene networks can be valuable to confirm previous knowledge and to derive new hypotheses. The meaning of edges can be various, ranging from transcriptional regulation to membership in the same operon or even more complex interactions, and has, therefore, to be assessed in the light of previous knowledge after gene network estimation. Since optimal gene network models show the truth from a certain, new perspective, it will be an important task to develop the understanding of such estimations further.

Figure 5.3: Extracted motif for *E. coli*.

Figure 5.4: Extracted motif for *B. subtilis*.

Chapter 6

Algorithms for Large Gene Networks

The methods described in the previous chapters can be applied to optimally estimate and enumerate gene networks for a limited number of genes. In this chapter, we introduce several approaches to also apply these methods when the number of genes exceeds the computational boundary.

We first introduce an approach to reduce the number of genes without losing essential information in Section 6.1. Applying this approach might be sufficient to reduce the number of genes to a feasible level in many research settings. In cases where this is not sufficient, we make use of the compartmentalisation of gene networks [Hartwell99, Milo02, Someren02] in order to decompose larger networks in smaller parts, and infer each partial network optimally. This means to first select a biologically relevant subspace of the search space. Then, we find optimal solutions in the selected subspace by repeatedly applying Algorithm 3 (page 40). Several approaches making use of this rationale are introduced in Section 6.2. The selection of the subspace can be done by applying biological knowledge or, if no previous knowledge is available, by using clustering methods or heuristic gene network estimation algorithms. The running time needed to find an optimal solution in the selected subspace is $O(n)$, where n is the number of genes, therefore allowing for the estimation of arbitrarily large gene networks. Since this approach allows for selecting and optimally searching a subspace that contains a very huge number of networks, this methodology is supposed to find optimal or nearly optimal solutions if the subspace is selected carefully. Even though compartmentalisation of gene networks is used, the interactions between compartments can be well incorporated with these techniques.

In Section 6.3, we show how Algorithm 4 (page 57) can be used for large gene networks as well in order to enumerate optimal networks in a subspace of the

search space. As a result, motif extraction techniques can be applied for large gene networks as well.

From the techniques proposed in this chapter, actual gene network estimators can be constructed in the following way. First, one should apply the method of Section 6.1 in order to remove redundant genes from the data set. Then, one of the approaches described in Section 6.2 should be selected and used in combination with the enumeration technique described in Section 6.3. Finally, a motif extraction method as described in Chapter 5 should be used to extract a reliable part from the enumerated network models. All algorithms of this chapter can be combined with each other in this way.

We present several applications of the algorithms introduced in this chapter in Section 6.4. Comparison of the estimated networks for *Bacillus subtilis* to biological knowledge yields a significant agreement.

6.1 Reducing the Number of Genes

When every gene in a group of genes $A \subseteq G$ shows the same or approximately the same pattern of gene expression, the gene network of this group can either not be infered from the data or is in fact the empty graph. In both cases, it is not reasonable to try a gene network estimation for A. Instead, the genes in A should be treated as a one element of the gene network, since it is reasonable to assume that they are regulated in the same way. We introduce the following annotation to capture this idea.

Definition 31:
We denote a group of genes that is biologically regulated in the same way as a *network element*. •

In idealised gene expression measurements, genes of network elements would be exactly the genes with correlation 1, but in real data the correlation will be lower than 1 because of measurement and system errors. However, the genes with highest correlation are very likely to be members of a network element, while genes with differential regulation are unlikely to show a higher correlation than genes that belong to the same network element.

We propose the following algorithm that makes use of this rationale. The algorithm takes one parameter $c \in]0, 1[$, which should be chosen close to 1.

Algorithm 6:

Step 1: Identify subsets $A_1, \ldots, A_n \subseteq G$ for which the correlation of g and h is at least c for all $g, h \in A_i$.

Step 2: For all $i = 1, \ldots, n$, average the gene expression measurements for genes in A_i.

Step 3: Use a gene network estimation method to estimate a gene network for $\{A_1, \ldots, A_n\}$.

This approach has three advantages compared to the application of a gene network estimator to the genes in G. First, since no differences of the regulation or the regulatory role of genes in a set A_i can be detected from the data, a gene network model for G would likely have many wrong edges and would be difficult to interpret. Second, the measurement error of gene expression measurements can be reduced by averaging. Third, the number of genes can be reduced, allowing faster estimation, easier graph drawing and a better understandability of the estimation result.

In our implementation of Algorithm 6, we use the Pearson correlation as defined on page 9. In order to identify network elements in Step 1, we apply hierarchical clustering [Johnson67]. Denoting the Pearson correlation of two genes as c, we use $1 - c$ as the distance measure for pairs of genes. We compute the distance of clusters as the minimum of the pairwise distances, which ensures correctness. While hierarchical clustering does not guarantee that all network elements can be detected, this approach worked well in application to real data (see Subsection 6.4.1).

6.2 Algorithms

We now discuss how to make use of biological knowledge to restrict the search space in a biologically slight but computationally effective way. It is a well-studied fact that the expression patterns of genes usually fall into groups of genes with similar pattern [Gasch00, Segal03, Spellman98], and therefore are not all individual. When such groups of genes are detected, the similarity of the expression patterns of the genes in one group make it likely that there will be many regulatory relationships within the group. On the contrary, the comparatively large difference of the gene expression patterns of genes in different groups make it unlikely that there will be as many regulatory relationships between genes of different groups. Therefore, it is to be expected that the real gene network will have dense components within groups, but will be comparatively sparse outside of groups.

In order to make use of this biological reality, we first give a theoretical fundament, and then show three approaches to apply it. All approaches guarantee optimality with respect to a restricted search space and therefore provide a clear distinction of the part of the estimation that is done heuristically/empirically, and the part that is done optimally.

6.2.1 Theoretical Basis

The following result prepares the ground for our approach.

Theorem 3 [Ott03]

Let m, $c \in \mathbb{N}$. For each $g \in G$, let $C_g \subseteq G - \{g\}$ be a set of candidate parents. Assume that the following two conditions hold:

1. *$|C_g| \leq m$ for all $g \in G$.*
2. *The strongly connected components of the graph induced by C_g, $g \in G$, are not larger than c.*

Then optimal networks with respect to the selected candidate parents can be found in time $O(|G|)$.

Proof. Let $L = (G, E)$ denote the graph on the set of genes G that is induced by C_g, $g \in G$. That means E contains exactly the edges (h, g) for which $h \in C_g$ holds. Let $S_1, \ldots, S_n \subseteq G$ denote the strongly connected components of L. Now let $L' = (\{S_1, \ldots, S_n\}, E')$ denote the graph on the strongly connected components with E' defined as

$$E' = \{(S_i, S_j) | i \neq j, \exists g \in S_i, h \in S_j : (g, h) \in E\}. \tag{6.1}$$

Then, by the definition of strongly connected components there is no cycle in L'. W.l.o.g. let us assume that $S_1, \ldots, S_n$ is a topological sort for L' which means that there is no edge (S_i, S_j) in L' with $j < i$.

Now let us show that Algorithm 3, with slight modifications, can be applied to compute optimal networks for G with respect to the selected candidate parents. In order to apply Algorithm 3 to a strongly connected component S_i, we modify the algorithm in the following way:

1. In the computation of F in Step 1 and Step 2, we only compute $F(g, A)$ for all $g \in S_i$ and all $A \subseteq C_g$.
2. We replace the term $F(g, A - \{g\})$ in Step 4a (see definition of Algorithm 3) by $F(g, (C_g - S_i) \cup (C_g \cap A))$.

These two changes introduce the restrictions for candidate parents to the algorithm, and allow $g \in S_i$ to have parents outside of S_i. This does not affect the correctness of Algorithm 3, since the proof for Theorem 1 (page 41) can be done for the modified Algorithm 3 in the same way as for Algorithm 3. We apply Algorithm 3 modified in this way to each strongly connected component S_i in an arbitrary order yielding partial networks $N_i = (G, E_i)$ for each $i \in 1, \ldots, n$. Then we return the network $N = (G, \bigcup_{i=1}^{n} E_i)$. By Theorem 1, each partial network N_i is optimal. From the acyclicity of L' it follows that N is acyclic. Furthermore, the selection of parents for a gene $g \in S_i$ can be done independently from the selection of parents in other components, since edges spanning two components can not be part of a cycle. Thus, using the definition $score(N) = \sum_{g \in G} s(g, Pa^N(g))$, N is an optimal network.

Since we have $|S_i| \leq c$ and $|C_g| \leq m$, the computation time for one call of Algorithm 3 becomes bound by some constant (Theorem 1). Therefore, applying Algorithm 3 to all strongly connected components can be done in $O(|G|)$, which completes the proof. $\triangle$

6.2.2 Using Clustering Methods

Clustering methods have worked well in several studies in order to identify functional groups of genes that show similarity in the pattern of gene expression [Gasch00, Segal03, Spellman98]. In the following, we show how clustering methods can be used in combination with Theorem 3 to identify a biological meaningful subspace of the search space and to make optimal estimations within the subspace.

6.2.2.1 Basic Approach

By Theorem 3, in order to allow for an effective network prediction, we need to select candidate parents for each gene such that the strongly connected components in the graph containing all candidate interactions can be bound by a constant. Since gene networks are assumed to consist of highly connected blocks, which are sparsely connected to each other [Hartwell99, Shen-Orr02], this restriction seems to be satisfiable in many research settings. Therefore, Theorem 3 provides the basis for a general approach of gene network estimation. We propose the following algorithm, which takes two parmeters $m, c \in \mathbb{N}$:

Algorithm 7: [Ott03]

Step 1: Cluster genes in G such that no cluster is larger than c genes.
Step 2: Sort the clusters by decreasing size: $C_1, \ldots, C_n$.
Step 3: For each $i \in \{1, \ldots, n\}$ and for each gene $g \in C_i$, select up to m candidate parents from $C_1 \cup \ldots \cup C_i$.
Step 4: Compute an optimal gene network model using Theorem 3.

Since the candidate parents for all genes g are chosen from the cluster g is contained in or previous clusters, no cycle in the graph induced by C_g, $g \in G$ can span two clusters. Therefore, the strongly connected components are not larger than c, because $|C_i| \leq c$ for all clusters C_i, which shows the correctness of Algorithm 7.

Genes that belong to the same cluster are correlated and should therefore form a densely connected part of the gene network, while genes that belong to different clusters have a lower correlation and are therefore unlikely to be connected in the gene network. We use k-means clustering [Hartigan79] with the Pearson correlation as distance measure, and the median to compute cluster centers.[1] Another possibility would be to use $\frac{1}{2} \cdot (s(g, \{h\}) + s(h, \{g\}))$ as a distance measure in combination with hierarchical clustering [Johnson67], which does not require the computation of cluster centers.

The rationale for sorting the clusters by decreasing size is as follows. The clusters at the beginning of the sorting have the strongest restriction for the selection of candidate parents, since there are few previous clusters. To maximise the number of genes, from which candidate parents can be selected, big clusters should be put at the beginning.

We note that the restrictions of the subspace imposed by Step 1 and Step 2 still allow any two genes to be connected, restricting only the direction of edges for some pairs of genes.

There are many reasonable criteria for the selection of a candidate parent h for a gene g in Step 3. Examples are the correlation of genes or $s(g, \{h\})$. In our implementations, we make use of the latter criteria.

6.2.2.2 Refined Approach

While Step 1 of Algorithm 7 is well justified as explained above, the sorting in Step 2 may potentially impose a restriction of the search space that does not contain all biologically plausable network models. As an alternative to the sorting step, we propose a refined approach in the following. In order to infer a biologically justified order of the clusters, we can compute the average of gene expression

[1] We made use of publicly available clustering software from [deHoon03c].

measurements within each cluster, and estimate an optimal gene network for the clusters.[2] This idea is used in the following algorithm, which also takes two parameters $m, c \in \mathbb{N}$:

Algorithm 8:

Step 1: Cluster genes in G such that no cluster is larger than c genes.
Step 2: Average the gene expression measurements in every cluster $C_1, \ldots, C_n$.
Step 3: Use Algorithm 3 to estimate an optimal network model N^* for $\{C_1, \ldots, C_n\}$.
Step 4: Compute a topological sorting for N^* to sort the clusters: $C_{j_1}, \ldots, C_{j_n}$.
Step 5: For each $i \in \{1, \ldots, n\}$ and for each gene $g \in C_{j_i}$, select up to m candidate parents from $C_{j_1} \cup \ldots \cup C_{j_i}$.
Step 6: Compute an optimal gene network model using Theorem 3.

As stated in Definition 8 (page 9), in a topological sorting of the vertices of an acyclic graph, all edges comply with the direction given by the sorting. Therefore, computing a topological sort in Step 3 assures that the sorting of clusters does not prohibit the selection of candidate parents from clusters that are predecessors in the optimal network computed in Step 2. Since an optimal network for the clusters should point to the gross structure of interactions between compartments, the selection of candidate parents in Step 5 should conform well to the structure of the real network. The correctness of Algorithm 8 follows in the same way as for Algorithm 7.

While in Algorithm 6 only gene expression measurements for genes with approximately the same pattern of gene expression were averaged, we average measurements for genes with similar, but potentially different gene expression pattern in Step 2. However, since the gene network estimated in Step 3 is used only to sort the clusters, this averaging procedure should not affect the gene network estimation strongly.

6.2.3 Improving Heuristic Results

While clustering is a promising approach for identifying biologically meaningful subspaces of the search space, there is also a different way to achieve this. Widely used heuristic algorithms like greedy algorithms [Imoto02] or simulated annealing [Hartemink02] do not guarantee finding optimal or approximately optimal

[2] Similar approaches were used in [deHoon03a] and [Mjolsness00], but no gene network estimation was done for the complete set of genes in these works.

solutions, but are likely to at least find solutions that are contained in the same subspace as optimal solutions. We make use of this assumption in the following algorithm, which takes two parameters $m, c \in \mathbb{N}$:

Algorithm 9:

Step 1: Use a heuristic algorithm to estimate a gene network N_H on G.
Step 2: Compute a topological sorting for N_H: $g_1, \ldots, g_n$.
Step 3: Group the genes in G in $m = \lceil \frac{|G|}{c} \rceil$ groups:
$C_i = \{g_{(i-1)\cdot c+1}, \ldots, g_{i\cdot c}\}$, $i = 1, \ldots, m-1$,
$C_m = \{g_{(m-1)\cdot c+1}, \ldots, g_n\}$.
Step 4: For each $i \in \{1, \ldots, m\}$ and for each gene $g \in C_i$, select up to m candidate parents from $C_1 \cup \ldots \cup C_i$.
Step 5: Compute an optimal gene network model using Theorem 3.

Again the correctness follows from the way of selection in Step 4. The parameters m and c can be used to adjust the size of the subspace around the heuristic solution in which an optimal solution is found in Step 5. However, the computation time increases as described in Chapter 3 with increasing m and c and must be kept in feasible limits. Since the genes are sorted topologically according to N_H, all parent genes in the heuristic solution can be selected as candidate parents in Step 4 in order to ensure that the heuristic solution is contained in the selected subspace. Since the selected subspace can contain a vast number of networks, as will be discussed in Section 6.3, this algorithm may find a global optimal solution even if the heuristic solution is only remotely related to the optimal solution.

6.2.4 Using Previous Knowledge

Algorithm 7, 8, and 9 make use only of the basic biological knowledge that gene networks can be decomposed in components of high connectivity. However, for many gene networks more specific biological knowledge is available. If there is previous knowledge concerning the densely connected components and the direction of the interaction of genes in different components, the following algorithm can be applied.

Algorithm 10: [Ott03]

Step 1: Group genes in G in groups C_i with $|C_i| \leq c$ and sort them according to biological knowledge: $C_1, \ldots, C_n$.
Step 2: For each $i \in \{1, \ldots, n\}$ and for each gene $g \in C_i$, select up to m candidate parents from $C_1 \cup \ldots \cup C_i$.
Step 3: Compute an optimal gene network model using Theorem 3.

As for Algorithms 7, 8, and 9, the correctness follows directly from the way candidate parents are chosen in Step 2. An example where not only the densely connected components, but also their order is known, are genes that are activated in specific phases of the cell cycle. In situations where the knowledge about densely connected components and/or their order is partial knowledge, a combination of Algorithm 7 or 8 and Algorithm 10 can be used.

6.3 Enumeration of Large Gene Networks

In this section we show that the enumeration of optimal networks can be done in subspaces of larger search spaces as defined in Theorem 3 as well. This allows to use motif extraction techniques as proposed in Chapter 5 for large gene networks in the same way as for small gene networks. We first reformulate the notion of a subspace as used in Theorem 3.

Definition 32:
Let $C_g \subseteq G - \{g\}$ for all $g \in G$. We define the *subspace induced by* C_g *for* $g \in G$ as $\{N \subseteq G \times G | N \text{ is acyclic}, \forall g \in G[Pa^N(g) \subseteq C_g]\}$. •

By the definition, a subspace is given by the selection of candidate parents for each gene. As described in the proof of Theorem 3, the strongly connected components of the graph induced by the candidate parents yield a decomposition of the set of genes G. Let us use $D_1, \ldots, D_n$ to denote this decomposition for a given subspace.

It was shown that Algorithm 3 (page 40) can be used iteratively to compute networks $N_i = (G, E_i)$ for each $i \in 1, \ldots, n$, when the set of genes G is decomposed in n parts. The resulting network for G is the network $N = (G, \bigcup_{i=1}^{n} E_i)$. In the same way as described in the proof of Theorem 3, we can modify Algorithm 4 (page 57) in order to account for gene interactions of genes in different components. Let us denote the modified version of Algorithm 4 as Algorithm 4′. We can apply Algorithm 4′ for the enumeration of a number of best networks for each of the n components. We can then construct networks for G from the enumerated partial networks. This idea is utilised in the following algorithm, which takes one parameter $m \in \mathbb{N}$.

Algorithm 11:

Step 1:	Select a subspace. The subspace induces a decomposition of G: $D_1, \ldots, D_n$.
Step 2:	Apply Algorithm 4′ to all D_i to find the best m networks $N_{i,k} = (G, E_{i,k})$, $k \leq m$, for D_i.
Step 3:	Set $\beta^{(1)} = (1, \ldots, 1) \in I\!N^n$.
Step 4:	For all $j = 2, \ldots, m$, do the following two steps:
Step 4a:	Select $i \leq n$ and $k < j$ minimising $score(N_{i,\beta_i^{(k)}}) - score(N_{i,\beta_i^{(k)}+1})$ among all i, k such that $(\beta_1^{(k)}, \ldots, \beta_{i-1}^{(k)}, \beta_i^{(k)}+1, \beta_{i+1}^{(k)}, \ldots, \beta_n^{(k)})$ does not match $\beta^{(p)}$ for $k \neq p < j$.
Step 4b:	Set $\beta_l^{(j)} = \beta_l^{(k)}$ for all $l \neq i$ and $\beta_i^{(j)} = \beta_i^{(k)} + 1$.
Step 5:	For all $j \leq m$, return $N_j = (G, \bigcup_{i=1}^n E_{i,\beta_i^{(j)}})$.

We now prove that Algorithm 11 does in fact enumerate the most likely networks within the selected subspace.

Theorem 4

Algorithm 11 finds the best m networks within the subspace selected in Step 1.

Proof. From Theorem 2 and Theorem 3 it follows that the networks computed in Step 2 are the best m networks for the respective components. Therefore, we only need to prove that the combinations of these partial networks as computed in Step 3 and Step 4, are the optimal solutions within the subspace. We prove this by induction over the index j of the networks returned in Step 5. First, since

$$\begin{aligned} N_1 &= (G, \textstyle\bigcup_{i=1}^n E_{i,\beta_i^{(1)}}) \\ &= (G, \textstyle\bigcup_{i=1}^n E_{i,1}) \end{aligned}$$

holds, it follows from Theorem 3 that N_1 is the most likely network within the subspace.

Now let us assume that for some $j \leq m$, we have that for all $i < j$, N_i is the i-th best network within the subspace. Let $M^* = (G, E^*)$ be a network with minimal score among the networks M for which $M \neq N_i$ holds for all $i < j$. Let $i < n$ and consider the partial network (G, E_i) with E_i defined as

$$E_i = \{(g, h) | h \in D_i\}. \tag{6.2}$$

From the minimality of $score(M^*)$ we see, that (G, E_i) must match $N_{i,k^{(i)}}$ for some $k^{(i)} < m$.[3] Now let α denote $(k^{(1)}, \ldots, k^{(n)})$. We prove the following three claims.

[3] As in Chapter 4, we assume that networks with equal score are ordered in some arbitrary order.

1. There exists $i < j$ such that $\beta_l^{(i)} \neq \alpha_l$ holds for exactly one $l \leq n$.
2. $\alpha_l > \beta_l^{(i)}$
3. $\alpha_l - \beta_l^{(i)} = 1$

For the first claim, let us assume it does not hold. Then, we can construct a network M' from M by replacing the partial network for one component by a better solution for this component. M' does not match N_i for $i < j$, which contradicts the minimality of $score(M)$. Therefore, the first claim holds.

For the second claim it suffices to observe that $\alpha_l \leq \beta_l^{(i)}$ would imply $\alpha = \beta^{(p)}$ for some $p < j$, since Algorithm 11 only increases one component of one previous vector in Step 4b. The third claim follows again from the the minimality of M. $\beta^{(j)}$ is chosen in Step 4b as a vector among vectors fulfilling above conditions that minimises the score difference of the network induced by it and the network induced by the nearest previous solution. We conclude $score(M) = score(N_j)$, which proves the theorem. △

We note that it may not be necessary to compute the m best networks for each component in order to find the best m networks within the subspace in practical applications. Similar to the combinatorial explosion used in Subsection 4.1.3, the combinatorial effect when forming a network from n partial networks will in practice allow more than m optimal combinations, even if less than m optimal networks for the single components are known. In order to utilise this effect, Algorithm 11 can be used with a slight modification. The loop in Step 4 must be terminated when one element of a computed vector $\beta^{(j)}$ is the rank of the last solution known for the particular component.

The size of the subspace as defined in Definition 32 is huge as we illustrate with one example. Let us assume dividing G in groups $D_1, \ldots, D_n$ of equal size q and select as candidate parents for a gene $g \in D_i$ the set $D_i - g$. Then the graph induced by the sets of candidate parents will be a collection of cliques without connections among each other. The number of networks in the subspace induced by this selection of candidate parents is c_q^n, where c_q denotes the number of directed acyclic graphs with q nodes (see page 33). Since in practical, still feasible applications c_q can be larger than 10^{70}, the subspaces can contain a vast number of networks. Moreover, when connections between components are considered, the calculation of the size of the subspace gets more complicated, but the size will only increase.

6.4 Applications and Results

As noted in the introduction to this chapter, gene network estimators can be composed from the algorithms in Sections 6.1, 6.2, and 6.3. The overall concept is depicted in Figure 6.1. When a gene network is to be estimated for a given set of genes G, the first step should be to detect network elements using Algorithm 6. Then a biologically meaningful subspace of the search space must be detected using Algorithm 7, 8, 9, or 10.[4] In the next step, we use Algorithm 3, 4, or 11 to find optimal solutions in the selected subspace. If we enumerate networks, we can further apply motif extraction algorithms like Algorithm 5 and output a gene network that is common to the likely networks of the selected subspace.

The reflexive edge in Figure 6.1 appended to "optimal search in subspace" indicates the possibility to iterate this step. After an optimal solution is found in the subspace, a new subspace can be constructed around this solution. This can be repeated until one finds a solution that is the optimal solution of the subspace around it.

We have implemented all algorithms introduced in this chapter and applied them to DNA microarray data. As score function s we chose the BNRC score [Imoto02], since it can model non-linear gene interactions and can handle the gene expression data without discretisation. Furthermore, we saw in Subsection 3.3.3 that the BNRC score yields results that are in better agreement with biological knowledge than estimations with other existing score functions.

The aim of this section is to evaluate and compare the predictive strength of the algorithms of this chapter.

6.4.1 Application of Algorithm 6

We selected a set of 100 *B. subtilis* genes, out of which 6 were sigma factors and 94 were genes known to be regulated by one the sigma factors [Sonenshein01]. We then used Algorithm 6 to estimate a gene network for this set of genes. Since sufficiently many and sufficiently large network elements could be detected (see below), we did not need to restrict to a subspace and used Algorithm 4 in Step 3 of Algorithm 6 to enumerate the 30 most likely gene network models, and extracted a gene network motif from these (result not shown).

Since the genes selected are active in various cellular responses and processes, we used the data set of 70 microarrays from time-course experiments with *B. subtilis* introduced in Subsection 4.3.1, which includes microarrays from various experiments like cold shock, phosphate limitation, or salt stress.[5]

[4] If the number of genes is within the feasible limit of Algorithm 3 discussed in Chapter 3, the selected subspace may be the whole search space.

[5] In Subsection 5.2.2, we used the opposite approach and selected genes that are known to play a role in tryptophan metabolism with microarray data specifically selected for this group.

Figure 6.1: Strategy for the estimation of gene networks.

The result of Step 1 of Algorithm 6 is shown in Table 6.1. As argued in Section 6.1, the groups of genes with highest pairwise correlation are likely to be true network elements as defined in Definition 31. Using the knowledge about operons and regulons from [Sonenshein01], we can confirm this from the result in Table 6.1. Among 30 predicted network elements, 10 are trivially correct, since they include only a single gene. In 11 groups all genes contained in the group are members of the same operon, and are, therefore, true network elements, since usually all genes in an operon are regulated in the same way. For the predicted network elements 2, 4, 6, 7, 8, 9, 10, 11, and 18 most genes belong to the same operon, but some genes from a different operon were placed in the same group. In all except one of these cases, the additional genes belong to the regulon of the same sigma factor. Therefore, it is likely that also these predictions are correct. For example, the genes of predicted network element 10 all belong to one large operon, except of *fliS* and *fliD*, which belong to a different operon. The one gene that was not member of the same operon or regulon is *mbl* in predicted network element 18, which is regulated by *sigE*, while *minD* and *minC* are regulated by *sigH*.

We conclude that nearly all predictions of network elements are correct. This allows to group many genes of the same network element in advance, therefore reducing noise by averaging and reducing the number of genes significantly without losing essential information.

6.4.2 Application of Algorithm 7 to *S. cerevisiae* data

We selected a data set of 33 microarrays, which where obtained in heat shock experiments (20 arrays) and osmotic shock experiments (13 arrays) [Gasch00] to *Saccharomyces cerevisiae*. Since these experiments should have affected the expression levels of genes that play a role in heat shock and/or osmotic shock responses, we selected a set of 38 such genes [Hohmann03]. We may expect that the data is to some degree informative with respect to the gene network of these genes. We applied Algorithm 7 to this input, limiting the size of clusters to 13 (parameter c), and the number of candidate parents to 10 (parameter m). The computation was performed using a single CPU with 1.9 GHz for about one hour. The result of the clustering step (Step 1 of Algorithm 7) is shown in Table 6.2, the estimated gene network is diagrammed in Figure 6.2.

We observe that the estimated gene network is a well connected graph. Therefore, even though a decomposition of the set of genes G is used to drastically reduce the computational burden, interactions of genes in different groups are well accounted for, such that the estimated gene network is not fractionised. Several genes play a role in the heat shock response as well as in the osmotic shock

Network element 1	spoIIIAH, spoIIIAG, spoIIIAF, spoIIIAB, spoIIIAA
Network element 2	spoIVFB, dacB, spoIVFA
Network element 3	glgD, glgC, glgA, glgB
Network element 4	mmgD, mmgC, mmgA, cotJC, cotJB, spoVID, cotJA
Network element 5	yjmF, yjmD, yjmG, yjmE, yjmC
Network element 6	flgK, flgM, tlpC, yvyG, yvyF
Network element 7	mcpB, mcpA, motB
Network element 8	fliI, fliG, fliH, fliF, lytD
Network element 9	fliM, flgE, fliK, fliJ, fliT
Network element 10	sigD, ylxL, fliL, ylxG, ylxF, flhB, fliR, fliP, fliZ, cheB, fliY,
	fliS, fliD, cheD, cheW, ylxH, flhF, flhA, fliQ, cheC
Network element 11	lytB, lytA, tlpB, lytC
Network element 12	racX, pbpE
Network element 13	yteJ, yteI
Network element 14	yknX, yknY
Network element 15	yuaF, yuaG
Network element 16	citG
Network element 17	ureB, ureA
Network element 18	minD, minC, mbl
Network element 19	ytxG, ytxJ, ytxH
Network element 20	lytR
Network element 21	psd, pssA, ybfM
Network element 22	dltB, dltA, dltC, dltD, dltE
Network element 23	katX
Network element 24	spoIIQ
Network element 25	lonB
Network element 26	sigE
Network element 27	sigW
Network element 28	sigH
Network element 29	sigX
Network element 30	sigF

Table 6.1: Predicted network elements.

response, but the heat shock factor *hsf1* and the transcription factor *hot1* can be assigned to the heat shock response respectively the osmotic stress response. Taking a closer look at these two genes, we see that all genes *hsf1* is estimated to interact with, are heat shock genes. Similarly, the estimated interaction partners of *hot1* are both genes that are involved in osmotic stress responses. Among the genes with the highest number of predicted interactions, we find two genes, namely *nth1* and *tps1*, with a metabolic function, and a putative heat shock gene of yet unknown function (*yro2*).

6.4.3 Application of Algorithm 7 to *B. subtilis* data

We have applied Algorithm 7 to *Bacillus subtilis* data (the time-course data set introduced in Subsection 4.3.1). We selected the DNA microarray data for 6 sigma factors and 79 operons known to be regulated by the chosen sigma factors [Sonenshein01]. The expression ratios of operons were defined as the average of the expression ratios of their respective genes (180 genes overall). Since we already

Cluster 1	yro2, smp1, ste50, ptc1, hig1, hsf1, sln1, ste11, msn1, ptp2
Cluster 2	ssa1, cdc24, hsp26, tps1, nth1, ptp3, pbs2, mcm1
Cluster 3	ubc4, hsp104, msn2, sko1, ssk2, hsp82
Cluster 4	ptc3, sod2, ssa2, ssk1, rck2, hot1
Cluster 5	ssa3, ptc2, msn4, cdc42
Cluster 6	gpd1, hxk1, ctt1
Cluster 7	ssk22

Table 6.2: Clustering used to narrow down the search space for a set of 38 yeast genes.

incorporated knowledge about network elements in this way, we did not make use of network element detection, and estimated a gene network for this set of 85 elements using Algorithm 7. Since we have 79 known regulatory interactions in this network of 85 genes respectively operons, we expect to re-discover a significant part of these, if Algorithm 7 has predictive power.

We set parameter m, the maximal number of selected candidate parents, to 15, and parameter c, the maximal size of clusters, to 25. The actual size of clusters computed in Step 1 ranged from 8 to 24. For this computation, we used a Sun Fire 15K supercomputer with 96 CPUs, 900 MHz each, for about one hour. Table 6.3 shows the clustering result of Step 1.

Cluster 1	sigE, sigF, spoIIP, spoIID, spoIIIAA, spoIIID, spoIVFA, cotJA dacB, spoVB, spoVD, spoVR, spoVID, glgB, mmgA, yteV, yrbA yaaH, cwlJ, yjmC, tlpA, spoVS, spoIIQ, lonB
Cluster 2	sigD, phoB, hag, yvyC, yvyF, motA, tlpB, tlpC, mcpA, mcpB, mcpC, flgB, lytA, lytD, yhdD, yoaF, ftsA
Cluster 3	sigW, ydbS, yeaA, yjoB, yknW, ywrE, yobJ, yfhL, yuaF, yvlA, xpaC, ythP, yqeZ, pssA
Cluster 4	yxjI, yteI, yndN, ygaI, citG, minC, ytxG, csbB, lytR, dltA, katX, spoIIR
Cluster 5	sigX, bofA, yknT, yoaW, cheV, degR, sigW(operon), yxzE, yoaG, rapD
Cluster 6	sigH, pbpE, ybfO, spoVG, spo0F, ureA, yvyD, kinA

Table 6.3: Clustering used to narrow down the search space for a set of 85 operons and genes.

In order to evaluate the biological significance of the estimated network, we computed the distance of each operon to each sigma factor in the network predicted by Algorithm 7, where we defined the distance as the minimal number of edges

on an undirected path connecting the operon and the sigma factor. When the correct sigma factor was closer to an operon than all other sigma factors, we rated the regulatory relationship as detected. This procedure allows to compute the p-values for our estimation result, since for a meaningless predicter the probability of detecting regulatory interactions would be at most $\frac{1}{6}$.[6]

Table 6.4 shows the result of this evaluation scheme. We observe that the estimated gene network is of significant agreement to the biological knowledge for two sigma factors, $sigE$ and $sigW$. This shows that the estimation result contains valuable biological information. We note that a majority of the undetected regulatory relationships were breaking ties, and therefore the distance of operons to their correct sigma factors was still minimal, but one or more of the other sigma factors was as close.

Sigma Factor	Detected Relations	Known Relations	p-value
sigE	8	22	0.021
sigD	5	16	0.113
sigW	12	21	$3.02 \cdot 10^{-5}$
sigH	1	11	0.865
sigX	0	5	1.0
sigF	0	4	1.0

Table 6.4: Application of Algorithm 7 to *Bacillus subtilis* data.

We repeated the computation using Algorithm 8 with the same parameters. The result is summarised in Table 6.5. In this case, the evaluation showed significant agreement of network estimation and knowledge for three sigma factors: $sigW$, $sigH$, and $sigX$. However, since the total number of operons of our data set that are known to be regulated by $sigX$ is very low, the significance for $sigX$ is not as firm as for the other two. We observe that the especially strong significance for $sigW$ is consistent with the result for Algorithm 7 and also with a result from [deHoon03b] for differential equation networks. Therefore, it is likely that the data set is more informative for the region of the gene network that is around $sigW$ than for other regions.

From the results in Table 6.4 and Table 6.5 we cannot judge which of both algorithms is stronger, since the k-means clustering procedure makes use of random initial clusterings, and therefore yielded a different clustering in the execution of both algorithms. The comparison of these two algorithms is the focus of the following subsection.

[6]The actual probability is lower than $\frac{1}{6}$, because we count breaking ties as undetected.

Sigma Factor	Detected Relations	Known Relations	p-value
sigE	6	22	0.147
sigD	4	16	0.271
sigW	10	21	$9.72 \cdot 10^{-4}$
sigH	5	11	0.025
sigX	3	5	0.035
sigF	0	4	1.0

Table 6.5: Application of Algorithm 8 to *Bacillus subtilis* data.

6.4.4 Comparison of Algorithms 7 and 8

As described above, Algorithm 7 as well as Algorithm 8 first compute a clustering to decompose the set of genes in smaller sets. Then these clusters are ordered, candidate parents are selected according to the ordering, and an optimal network within the selected subspace is computed as described in the proof of Theorem 3. The only difference of Algorithm 7 and Algorithm 8 is the way the ordering of clusters is selected, both making use of a different rationale. Algorithm 7 orders the clusters by size in order to maximise the number of parents from which candidate parents can be selected for the first clusters. Algorithm 8 computes an optimal network with the clusters as elements in order to explore the overall network structure, and to select an ordering of clusters that complies with this structure.

The goal of this subsection is to answer the question which of these two rationales is stronger in order to find gene networks that agree with biological knowledge. In order to do so, we selected the unpublished data set of 99 *Bacillus subtilis* DNA microarrays obtained from disruptant experiments (see Subsection 4.3.1). Then we selected 100 random target networks of 60 genes each applying the random procedure described in Subsection 4.3.2 (page 68). For each target network, we applied both algorithms with parameters $c = 12$ and $m = 12$ to estimate gene networks. The accuracy of the resulting 200 gene network models was evaluated by the evaluation scheme described in Subsection 5.2.1 (page 78) making use of knowledge from the DBTBS [Makita03]. The result is summarised in Table 6.6.

We observe that randomly selected edges from estimations of Algorithm 8 were correct more often than for Algorithm 7. This holds for correctness in the sense of Subsection 4.3.2 as well as for the strictest form of correctness that only accepts predicted interactions that match database entries exactly. These results

Algorithm	Number of Hits	p-value	Number of Hits Strict	p-value Strict
Algorithm 7	25	0.131	8	$2.71 \cdot 10^{-3}$
Algorithm 8	46	$4.22 \cdot 10^{-9}$	9	$6.50 \cdot 10^{-4}$

Table 6.6: Summarised estimation result for 100 target networks of 60 genes each. One edge is selected randomly from each estimated network and judged as true, if it can be justified with the biological knowledge respectively if it matches a database entry exactly (strict criteria).

indicate that the concept of Algorithm 8 to first investigate the overall structure of the gene network is superior to the sorting strategy of Algorithm 7. In the case of the relaxed criteria, the significance of estimations by Algorithm 7 is not even certain, though we must consider that the given values are upper bounds for the true p-values which might be much lower (see Subsection 5.2.1 on page 78). However, in the case of the strict criteria, the difference of the prediction accuracy of both algorithms is not significant. This might be explained by the relatively low number of 100 iterations, considering the sparse knowledge the strict criteria provides.[7]

6.4.5 Application of Algorithm 9

Algorithm 9 uses an estimated gene network N as a starting point, and optimally searches a huge subspace of networks built around N. In this subsection, we investigate, whether the improvement that can be achieved by this search is significant.

As in the previous subsection, we make use of *Bacillus subtilis* microarray data from disruptant experiments, knowledge from the DBTBS, the random selection procedure for target networks from Subsection 4.3.2, and the evaluation scheme described and applied in Subsection 5.2.1. We selected 300 target networks of 60 genes each, applied Algorithm 2 (page 34), Algorithm 7, and Algorithm 8 to make three gene network estimations for each target network. Each estimated network was then used as a starting point for Algorithm 9.[8] The number of greedy iterations for Algorithm 2 was set to 200, parameters c and m for Algorithms 7, 8, and 9 were set to 10.

The result of this extensive computation is summarised in Table 6.7 for the criteria that also accepts edges that are deducible from the knowledge, while Table 6.8 shows the result for the strict criteria.

[7] By the strict criteria not more than an average of about 2% of the possible edges are considered as correct edges. Therefore, many biologically correct estimations will be judged as wrong, if the strict criteria is applied.

[8] In the case of combining Algorithm 7 respectively Algorithm 8 with Algorithm 9, this corresponds to the reflexive edge drawn in Figure 6.1.

Algorithm	Number of Hits	p-value	Number of Hits with Algorithm 9	p-value with Algorithm 9
Algorithm 2	121	$6.37 \cdot 10^{-16}$	135	$1.34 \cdot 10^{-22}$
Algorithm 7	106	$4.67 \cdot 10^{-10}$	104	$2.23 \cdot 10^{-9}$
Algorithm 8	126	$3.55 \cdot 10^{-18}$	107	$2.09 \cdot 10^{-10}$

Table 6.7: Evaluation of the improvement achieved by the application of Algorithm 9.

Algorithm	Number of Hits	p-value	Number of Hits with Algorithm 9	p-value with Algorithm 9
Algorithm 2	16	$1.73 \cdot 10^{-3}$	26	$1.06 \cdot 10^{-8}$
Algorithm 7	17	$6.58 \cdot 10^{-4}$	20	$2.56 \cdot 10^{-5}$
Algorithm 8	23	$6.35 \cdot 10^{-7}$	22	$2.28 \cdot 10^{-6}$

Table 6.8: Evaluation as in Table 6.7 using the strict acceptance criteria.

We observe a significant improvement of estimation accuracy in the case of improving solutions computed with Algorithm 2, while there was no significant improvement in the case of the other two algorithms. This might be explained by the similar technique employed in Algorithm 9 as well as in Algorithm 7 and Algorithm 8. The stronger improvement in the case of Algorithm 2 which uses a very different approach might indicate that strong and weak points of Algorithm 2 and Algorithm 9 complement each other. We note that in all cases the gene network estimated by Algorithm 9 had a better score than the starting points computed by the other three algorithms. Therefore, the lack of improvement of estimation accuracy in some cases is an indication that the score function itself or its parameters might be improvable. Since we observe significant improvements with respect to the widely used Greedy Hill Climbing (Algorithm 2) [Friedman98b, Heckerman95, Imoto02, Kim03, Nariai04, Tamada03], Algorithm 9 should be used in order to improve gene network estimations in bioinformatics research.

In an application of Algorithm 7 and Algorithm 8 for the purpose of biological or medical research, one would take the opportunity to inspect the clustering of genes used by these algorithms by eye in the light of previous knowledge and/or experience. If, for example, the clustering falls into too many small clusters, these could be merged in order to improve the gene network estimations. Similarly, if a rather large cluster of genes is identified, it might be decided to not further divide it, and to spend more computation time for its processing. However, in the above

evaluation the clustering was done automatically without human interference, and therefore the predictive power of these algorithms might be even higher if applied more carefully.

6.4.6 Application of Algorithm 10 to yeast cell cycle data

Algorithm 10 is based on the application of previous knowledge in order to find a biologically meaningful subspace. Here, we apply Algorithm 10 to DNA microarray data for *Saccharomyces cerevisiae* obtained from time-course experiments using synchronised cell cultures [Spellman98]. This data set can be expected to contain some information about the gene network that works during the cell cycle of yeast.

It is known that the three gene pairs *cln1/cln2*, *clb5/clb6*, and *clb1/clb2* are activated specificly during the G1/S phase, the S phase, respectively the M phase of the cell cycle [Cho98]. By a clustering analysis based on this fact, several genes were predicted to be predominantly activated during one of these phases [Nariai04]. We selected a set of 43 genes, divided them into three groups, each group corresponding to one part of the cell cycle. We sorted these groups according to the time order of the cell cycle phases (Step 1 of Algorithm 10), set the maximum number of candidate parents (parameter m) to 20, and applied Algorithm 10 to estimate a gene network. The computation was done using a single CPU with 1.9 GHz for less than one hour. Figure 6.3 shows the estimation result.

We observe a total number of 87 predicted gene interactions. Table 6.9 shows the number of directed interactions from a gene g to a gene h partioned by the groups g (rows) and h (columns) belong to. We see that 52 out of 87 interactions are within groups, well reflecting the expectation that most interactions occur between genes that are predominantly active during the same phase of the cell cycle. 35 interactions are estimated for pairs of genes from different groups. The number of interactions between two groups is 13 for each of the temporal neighbours G1/S and S/G2, respectively S/G2 and M, but only 9 for the interactions from G1/S phase to M phase. Therefore, though the gene network estimation is based on a decomposition of the set of genes in three groups, interactions between groups are well accounted for.

We counted the number of interactions for each gene g, which is the number of parents of g plus the number of children of g. The genes with the highest number of predicted interactions in the estimated gene network (Figure 6.3) are listed in Table 6.10, which also gives the molecular function of these genes as annotated by the Saccharomyces Genome Database [SGD]. We observe that all genes with at least 7 predicted interactions have regulatory activity on the transcriptional

9	13	18	M
13	16		S/G2
18			G1/S
G1/S	S/G2	M	

Table 6.9: Number of predicted interactions by gene groups.

level or have a yet unknown function. Among the three genes with 6 predicted interactions one gene (*hcm1*) has regulatory function on the transcriptional level, while the two others are cell cycle dependent signalling proteins. We conclude that all genes with a higher number of predicted interactions have a known active regulatory role or are of unknown function. This shows that the estimated gene network contains at least some degree of biological information.

The observation that the important genes *cln2* and *clb5* have a lower rank within the group of genes in Table 6.10 may be explained by their role in signalling which is observable from mRNA measurements only in an indirect way. Therefore, the activity of the other genes on the transcriptional level is expected to be more easily observable.

Gene	Number of Interactions	Molecular Function
yfr012w	8	unknown
yox1	7	DNA binding, specific transcriptional repressor activity
stb1	7	transcriptional activator activity
htb2	7	DNA binding
hek2	7	mRNA binding
swi5 [9]	7	transcriptional activator activity
ydr149c	7	unknown
hcm1	6	specific RNA polymerase II transcription factor activity
cln2	6	cyclin-dependent protein kinase, intrinsic regulator activity
clb5	6	cyclin-dependent protein kinase, intrinsic regulator activity

Table 6.10: Molecular functions of genes with at least 6 predicted interactions.

Table 6.11 shows on the contrary the genes with the lowest number of predicted interactions. We find several genes with metabolic or signalling functions, one gene of unknown function, one mRNA binding gene (*sto1*), and one transcription

[9] Interestingly, *swi5* is predicted to be regulated by *fkh2* which is a known regulatory relation [Zhu00].

factor (*swi4*). Since *sto1* is known to be involved in nuclear splicing, a regulatory role is less likely. Therefore, only one out of the 11 genes with lowest number of predicted interactions has a known regulatory function on the transcriptional level, which is biologically plausible, since many signalling events will not be observable from mRNA measurements. We conclude that gene network estimations can give more insight to the function of genes than mere clustering results.

Gene	Number of Interactions	Molecular Function
sco2	1	unknown
sto1	1	mRNA binding
bbp1	2	structural constituent of cytoskeleton
sur1	2	transferase activity, transferring glycosyl groups
clb3	2	cyclin-dependent protein kinase, intrinsic regulator activity
clb6	2	cyclin-dependent protein kinase, intrinsic regulator activity
adh2	2	alcohol dehydrogenase activity
ino80	2	ATPase activity
swi4	2	transcription factor activity
smc1	2	AT DNA binding, ATPase activity, DNA secondary structure binding, double-stranded DNA binding
smc3	2	ATPase activity

Table 6.11: Molecular functions of genes with at most 2 predicted interactions.

Figure 6.2: Estimation result of Algorithm 7 applied to heat shock genes and osmotic stress genes.

Figure 6.3: Gene network predicted by Algorithm 10. Columns represent parents in the network, rows children.

Chapter 7

Conclusion and Outlook

We have proposed a method that allows for optimally estimating gene networks of up to about 20–35 network elements, depending on whether the number of parents of each gene is assumed to be limited by some constant or not. This makes it possible to compare the estimation accuracy of different score functions, to assess the best parameters for a given scoring scheme, and to evaluate the usefulness of given microarray data, since optimal solutions are obtained. Also, the method is especially useful in settings where researchers focus on a certain group of genes and want to exploit gene expression measurements concerning these genes to the full extent. In contrast to heuristic approaches, it can be concluded that the statistical model is not adequate or the data is insufficient, if the estimation results are unsatisfying or contradictory to biological knowledge. Even for a network of 20 network elements, getting to know the best network from the huge search space given is a large amount of information.

Furthermore, we have extended this algorithmic approach to allow for the enumeration of optimal and suboptimal gene network models, and applied and evaluated methods for the extraction of common parts of likely networks. We have also used these techniques to do the first comparison of gene networks among related species based on estimations.

Finally, we have provided a theoretical basis for a general approach to estimate large gene networks efficiently. We have shown that this approach allows for estimating meaningful gene networks, and for the application of biological knowledge appropriately and effectively, while the computation time does not impose practical limitations for the number of genes. Enumeration of most likely gene networks can be done in this case as well, so that the approaches of finding gene network motifs are not limited to small gene networks.

The algorithmic methology described in this thesis is not dependent on a certain scoring scheme or a certain kind of gene expression measurements. Therefore,

these techniques are generally applicable in all situations, where a score function s with the functionality $s : G \times 2^G \to \mathbb{R}$ is given, which is the case for all scores within the Bayesian network framework, but also for most other score functions. This is an important property for gene network estimation techniques, since work on new scores incorporating previous knowledge such as sequence information [Tamada03] or protein interaction data [Imoto03a, Ito01, Nariai04] is on-going.

Rigorously assessing the accuracy of gene network estimations by comparison to available knowledge, as we conducted in Section 4.3, Subsection 5.2.1, and Section 6.4 is a promising approach to develop standards for assessing the strength of gene network estimation methods. This should be further pursued in the future.

Use for Biological Research

The focus of this thesis is the development of gene network estimation techniques and the validation of their usefulness. Though the proposed techniques have the potential to allow for new biological insights, this has not been conducted intensively in this work. When more gene expression data becomes available in the future, the data might carry precious pieces of new information. Since the opportunity to gain such information should not be missed, we list some important steps that should be part of any thorough gene network estimation, but were not performed in a majority of the automated evaluations of this thesis, because human supervision is required (see also Figure 6.1).

First, gene network element detection should be employed. The acceptance threshold has to be adjusted to the data set, since pairs of genes of the same network element show different magnitudes of correlation, depending on the type of experiments conducted. Also previous knowledge is useful to check the correctness of the detected network elements. Detection of network elements contributes by reducing the measurement noise through averaging and avoiding meaningless estimations among genes with nearly identical pattern of gene expression. Second, the selection of the set of target genes G should be done under consideration of the type of experiments and previous knowledge about the function of genes (this was exemplified in Subsection 5.2.2). The more a gene network was activated during experiments, the better estimation results can be expected. Third, previous knowledge should be used to the full extent. This applies to the definition of the prior probability as well as to the analysis of estimation results [Ideker02] (see also Subsection 5.2.2). If all these steps are carried out, it can be expected that the estimation accuracy is even higher than the accuracy we observed in our automated evaluations.

Estimating gene networks from gene expression measurements can only provide information about the structure of the gene network, but hardly information about the nature of observed gene interactions. However, this might be a powerful

property, since many different kinds of interactions can be detected with the same technique.

Implications for the Design of Microarray Experiments

Since for gene networks of up to about 35 network elements (corresponding to about 100 genes) optimal gene network models can be computed, it would be promising to design microarrays with only a few hundred well-selected genes, and to do experiments with such arrays in a high quantity in order to achieve the best possible result for the group of genes under consideration.

Currently it is not known which of the experimental techniques like gene disruption, gene overexpression, various shock conditions, hormone treatment, or time-course experiments are the most effective for the elucidation of gene networks. It might depend on the target network itself, and also combinations of different techniques might turn out to be the strongest approach [Miyano03]. In order to answer these questions in the future, the computation of optimal gene network models based on different kinds of data will be a helpful contribution (exemplified in 5.2.1.3). Also, principled evaluation of gene network estimations will be important. Iterating the cycle of experiment, estimation, and conclusion might be of interest as well [Gardner03, Ideker00].

Future Directions

One primary target of gene network estimations should be the identification of proteins as drug targets [Lengauer02] by revealing interactions with other genes with a known role in disease. Another medical application might be the analysis of individual differences of gene networks with the objective of designing and administering medicine adjusted to individuals.

When a sufficient amount of data becomes available, another direction of future research can be the analysis of significantly common gene network motifs (network motif in the sense of [Milo02]) in estimated gene networks in order to reveal the basic building blocks of gene networks [Bhan02]. Prior analyses have often been limited to the sparse knowledge about gene networks [Shen-Orr02, Thieffry98].

There are indications that evolution may be driven more strongly by changes in expression levels than changes in protein structures [Enard02, Oleksiak02]. Therefore, gene network estimation techniques might become valuable for evolutionary studies based on gene expression measurements.

Bibliography

[Akaike73] H. Akaike. Information theory and an extension of the maximum likelihood principle. *International Symposium on Information Theory*, **2**: 267–281, 1973.

[Akutsu99] T. Akutsu, S. Miyano, S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. *Pacific Symposium on Biocomputing*, **4**: 17–28, 1999.

[Akutsu00a] T. Akutsu, S. Miyano, S. Kuhara. Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, **16**: 727–734, 2000.

[Akutsu00b] T. Akutsu, S. Miyano, S. Kuhara. Algorithms for identifying Boolean networks and related biological networks based on matrix multiplication and fingerprint function. *Journal of Computational Biology*, **7**: 331–343, 2000.

[Bellman62] R. Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the Association for Computing Machinery*, **9**: 61–63, 1962.

[Bhan02] A. Bhan, D.J. Galas, T.G. Dewey. A duplication growth model of gene expression networks. *Bioinformatics*, **18**: 1486–1493, 2002.

[deBoor78] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, Berlin, 1978.

[Bouckaert94] R.R. Bouckaert. Properties of Bayesian network learning algorithms. *Conference on Uncertainty in Artificial Intelligence*, **10**: 102–109, 1994.

[Buntine91] W. Buntine. Theory refinement on Bayesian networks. *Conference on Uncertainty in Artificial Intelligence*, **7**: 52–60, 1991.

[Chen99] T. Chen, H.L. He, G.M. Church. Modeling gene expression with differential equations. *Pacific Symposium on Biocomputing*, **4**: 29–40, 1999.

[Chickering96] D.M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher, H.-J. Lenz (Eds.), *Learning from Data: Artificial Intelligence and Statistics V*, Springer-Verlag, pp. 121–130, 1996.

[Cho98] R.J. Cho, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockhart, R.W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, **2**: 65–73, 1998.

[Cooper92] G.F. Cooper, E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, **9**: 309–347, 1992.

[Courcelle01] J. Courcelle, A. Khodursky, B. Peter, P.O. Brown, P.C. Hanawalt. Comparative gene expression profiles following UV exposure in wild-type and SOS-deficient *Escherichia coli*. *Genetics*, **158**: 41–64, 2001.

[Cover91] T.M. Cover, J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.

[Davison86] A.C. Davison. Approximate predictive likelihood. *Biometrika*, **73**: 323–332, 1986.

[DeGroot70] M.H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

[D'haeseleer99] P. D'haeseleer, X. Wen, S. Fuhrman, R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. *Pacific Symposium on Biocomputing*, **4**: 41–52, 1999.

[Enard02] W. Enard, P. Khaitovich, J. Klose, S. Zöllner, F. Heissig, P. Giavalisco, K. Nieselt-Struwe, E. Muchmore, A. Varki, R. Ravid, G.M. Doxiadis, R.E. Bontrop, S. Pääbo. Intra- and interspecific variation in primate gene expression patterns. *Science*, **296**: 340–343, 2002.

[Friedman96] N. Friedman, Z. Yakhini. On the sample complexity of learning Bayesian networks. *Conference on Uncertainty in Artificial Intelligence*, **12**: 274–282, 1996.

[Friedman98a] N. Friedman, K. Murphy, S. Russell. Learning the structure of dynamic probabilistic networks. *Conference on Uncertainty in Artificial Intelligence*, **14**: 139–147, 1998.

[Friedman98b] N. Friedman, M. Goldszmidt. Learning Bayesian networks with local structure. In M.I. Jordan (Ed.), *Learning in Graphical Models*, Kluwer Academic Publishers, pp. 421–459, 1998.

[Friedman00] N. Friedman, M. Linial, I. Nachman, D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, **7**: 601–620, 2000.

[Friedman03] N. Friedman, D. Koller. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, **50**: 99–125, 2003.

[Gardner03] T.S. Gardner, D. di Bernardo, D. Lorenz, J.J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, **301**: 102–105, 2003.

[Garey79] M.R. Garey, D.S. Johnson. *Computers and Intractability*. W.H. Freeman and Company, 1979.

[Gasch00] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, P.O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, **11**: 4241–4257, 2000.

[Glover98] J.R. Glover, S. Lindquist. Hsp104, Hsp70, and Hsp40: a novel chaperone system that rescues previously aggregated proteins. *Cell*, **94**: 73–82, 1998.

[Gollnick02] P. Gollnick, P. Babitzke, E. Merino, C. Yanofsky. Aromatic amino acid metabolism in *Bacillus subtilis*. In A.L. Sonenshein, J.A. Hoch, R. Losick (Eds.), *Bacillus subtilis and its Closest Relatives: From Genes to Cells*, American Society for Microbiology, Washington, D.C., pp. 233–244, 2002.

[Hartemink01] A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, R.A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Pacific Symposium on Biocomputing*, **6**: 422–433, 2001.

[Hartemink02] A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, R.A. Young. Combining location and expression data for principled discovery of genetic regulatory network models. *Pacific Symposium on Biocomputing*, **7**: 437–449, 2002.

[Hartigan79] J.A. Hartigan, M.A. Wong. A k-means clustering algorithm. *Applied Statistics*, **28**: 100–108, 1979.

[Hartwell99] L.H. Hartwell, J.J. Hopfield, S. Leibler, A.W. Murray. From molecular to modular cell biology. *Nature*, **402**: C47–C52, 1999.

[Hastie90] T. Hastie, R. Tibshirani. *Generalized Additive Models*. Chapman & Hall, London, 1990.

[Heckerman95] D. Heckerman, D. Geiger, D.M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, **20**: 197–243, 1995.

[Heckerman98] D. Heckerman. A tutorial on learning with Bayesian networks. In M.I. Jordan (Ed.), *Learning in Graphical Models*, Kluwer Academic Publishers, pp. 301–354, 1998.

[Henner93] D. Henner, C. Yanofsky. Biosynthesis of aromatic amino acids. In A.L. Sonenshein, J.A. Hoch, R. Losick (Eds.), *Bacillus subtilis and Other Gram-positive Bacteria: Biochemistry, Physiology and Molecular Genetics*, American Society for Microbiology, Washington, D.C., pp. 269–280, 1993.

[Hohmann03] S. Hohmann, W.H. Mager (Eds.). *Yeast Stress Responses*. Springer-Verlag, Berlin, 2003.

[Holter01] N.S. Holter, A. Maritan, M. Cieplak, N.V. Fedoroff, J.R. Banavar. Dynamic modeling of gene expression data. *Proceedings of the National Academy of Sciences USA*, **98**: 1693–1698, 2001.

[Hooke1665] R. Hooke. *Micrographia*. London, 1665.

[deHoon03a] M.J.L. de Hoon, S. Imoto, K. Kobayashi, N. Ogasawara, S. Miyano. Inferring gene regulatory networks from time-ordered gene expression data of *Bacillus subtilis* using differential equations. *Pacific Symposium on Biocomputing*, **8**: 17–28, 2003.

[deHoon03b] M.J.L. de Hoon, S. Ott, S. Imoto, S. Miyano. Validation of noisy dynamical system models of gene regulation inferred from time-course gene expression data at arbitrary time intervals. *manuscript*.

[deHoon03c] M.J.L. de Hoon, S. Imoto, J. Nolan, S. Miyano. Open source clustering software. *Bioinformatics*, in press, 2003.
`http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/`

[Ideker00] T. Ideker, V. Thorsson, R.M. Karp. Discovery of regulatory interactions through perturbation: inference and experimental design. *Pacific Symposium on Biocomputing*, **5**: 302–313, 2000.

[Ideker02] T. Ideker, O. Ozier, B. Schwikowski, A.F. Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, **18**: 233–240, 2002.

[Imoto02] S. Imoto, T. Goto, S. Miyano. Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. *Pacific Symposium on Biocomputing*, **7**: 175–186, 2002.

[Imoto03a] S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, S. Miyano. Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks. *Computational Systems Bioinformatics*, **2**: 104–113, 2003.

[Imoto03b] S. Imoto, S. Kim, T. Goto, S. Aburatani, K. Tashiro, S. Kuhara, S. Miyano. Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network. *Journal of Bioinformatics and Computational Biology*, **1**: 231–252, 2003.

[Ishii01] T. Ishii, K. Yoshida, G. Terai, Y. Fujita, K. Nakai. DBTBS: A database of *Bacillus subtilis* promoters and transcription factors. *Nucleic Acids Research*, **29**: 278–280, 2001.

[Ito01] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences USA*, **97**: 4569–4574, 2001.

[Johnson67] S.C. Johnson. Hierarchical clustering schemes. *Psychometrika*, **2**: 241–254, 1967.

[Khodursky00a] A.B. Khodursky, B.J. Peter, M.B. Schmid, J. DeRisi, D. Botstein, P.O. Brown. Analysis of topoisomerase function in bacterial replication fork movement: Use of DNA microarrays. *Proceedings of the National Academy of Sciences*, **97**: 9419–9424, 2000.

[Khodursky00b] A.B. Khodursky, B.J. Peter, N.R. Cozzarelli, D. Botstein, P.O. Brown, C. Yanofsky. DNA microarray analysis of gene expression in response to physiological and genetic changes that affect tryptophan metabolism in *Escherichia coli*. *Proceedings of the National Academy of Sciences*, **97**: 12170–12175, 2000.

[Kim03] S.-Y. Kim, S. Imoto, S. Miyano. Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinformatics*, **4**: 228–235, 2003.

[Lee02] T.I. Lee, N.J. Rinaldi, F. Robert, D.T. Odom, Z. Bar-Joseph, G.K. Gerber, N.M. Hannett, C.T. Harbison, C.M. Thompson, I. Simon, J. Zeitlinger, E.G. Jennings, H.L. Murray, D.B. Gordon, B. Ren, J.J. Wyrick, J.-B. Tagne, T.L. Volkert, E. Fraenkel, D.K. Gifford, R.A. Young. Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science*, **298**: 799–804, 2002.

[Lengauer02] T. Lengauer (Ed.). *Bioinformatics - from Genomes to Drugs*. Wiley-VCH Verlag GmbH, 2002.

[Liang98] S. Liang, S. Fuhrman, R. Somogyi. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing*, **3**: 18–29, 1998.

[Lin02] D.-I. Lin, Z.M. Kedem. Pincer-search: an efficient algorithm for discovering the maximum frequent set. *IEEE Transactions on Knowledge and Data Engineering*, **14**: 553–566, 2002.

[Makita03] Y. Makita, M. Nakao, N. Ogasawara, K. Nakai. DBTBS: database of transcriptional regulation in *Bacillus subtilis* and its contribution to comparative genomics. *Nucleic Acids Research*, **32**: D75-D77, 2004.

[Martinez-Y.00] M.A. Martinez-Yamout, G. Legge, O. Zhang, P.E. Wright, H.J. Dyson. Solution structure of the cysteine-rich domain of *Escherichia coli* chaperone protein DnaJ. *Journal of Molecular Biology*, **300**: 805–818, 2000.

[Matsuno00] H. Matsuno, A. Doi, M. Nagasaki, S. Miyano. Hybrid Petri net representation of gene regulatory network. *Pacific Symposium on Biocomputing*, **5**: 338–349, 2000.

[Maxam77] A.M. Maxam, W. Gilbert. A new method for sequencing DNA. *Proceedings of the National Academy of Sciences USA*, **74**: 560–564, 1977.

[Milo02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon. Network motifs: simple building blocks of complex networks. *Science*, **298**: 824–827, 2002.

[Miyano03] S. Miyano. Gene network inference and biopathway modeling. *Journal of Bioinformatics and Computational Biology*, in press, 2003.

[Mjolsness00] E. Mjolsness, T. Mann, R. Castaño, B. Wold. From coexpression to coregulation: an approach to inferring transcriptional regulation among gene classes from large-scale expression data. *Advances in Neural Information Processing Systems*, **12**: 928–934, 2000.

[Nariai04] N. Nariai, S. Kim, S. Imoto, S. Miyano. Using protein-protein interactions for refining gene networks estimated from microarray data by Bayesian networks. *Pacific Symposium on Biocomputing*, **9**: 336–347, 2004.

[Nussinov78] R. Nussinov, G. Pieczenk, J.R. Griggs, D.J. Kleitman. Algorithms for loop matchings. *SIAM Journal of Applied Mathematics*, **35**: 68–82, 1978.

[Oleksiak02] M.F. Oleksiak, G.A. Churchill, D.L. Crawford. Variation in gene expression within and among natural populations. *Nature Genetics*, **32**: 261–266, 2002.

[Omnibus] `http://www.ncbi.nlm.nih.gov/geo/`

[Ong02] I.M. Ong, J.D. Glasner, D. Page. Modelling regulatory pathways in *E. coli* from time series expression profiles. *Bioinformatics*, **18**: 241–248, 2002.

[Ott03] S. Ott, S. Miyano. Finding optimal gene networks using biological constraints. *Genome Informatics*, **14**: 124–133, 2003.

[Ott04a] S. Ott, S. Imoto, S. Miyano. Finding optimal models for small gene networks. *Pacific Symposium on Biocomputing*, **9**: 557–567, 2004.

[Ott04b] S. Ott, A. Hansen, S.-Y. Kim, S. Miyano. Superiority of network motifs over optimal networks and an application to the revelation of gene network evolution. *submitted for publication.*

[Pe'er01] D. Pe'er, A. Regev, G. Elidan, N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, **17**: 215–224, 2001.

[Rissanen89] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, River Edge, N.J., 1989.

[Robinson73] R.W. Robinson. Counting labeled acyclic digraphs. In F. Harary (Ed.), *New Directions in the Theory of Graphs*, Academic Press, New York, pp. 239–273, 1973.

[Rung02] J. Rung, T. Schlitt, A. Brazma, K. Freivalds, J. Vilo. Building and analysing genome-wide gene disruption networks. *Bioinformatics*, **18**: 202–210, 2002.

[Salgado01] H. Salgado, A. Santos-Zavaleta, S. Gama-Castro, D. Millán-Zárate, E. Díaz-Peredo, F. Sánchez-Solano, E. Pérez-Rueda, C. Bonavides-Martínez, J. Collado-Vides. RegulonDB (version 3.2): transcriptional regulation and operon organization in *Escherichia coli* K-12. *Nucleic Acids Research*, **29**: 72–74, 2001.

[Sanger75] F. Sanger, A.R. Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of Molecular Biology*, **94**: 441–448, 1975.

[Schwarz78] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, **6**: 461–464, 1978.

[Segal03] E. Segal, R. Yelensky, D. Koller. Genome-wide discovery of transcriptional modules from DNA sequence and gene expression. *Bioinformatics*, **19**: 273–282, 2003.

[SGD] `http://www.yeastgenome.org/`

[Shen-Orr02] S.S. Shen-Orr, R. Milo, S. Mangan, U. Alon. Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nature Genetics*, **31**: 64–68, 2002.

[Shi98] Y. Shi, D.D. Mosser, R.I. Morimoto. Molecular chaperones as HSF1-specific transcriptional repressors. *Genes & Development*, **12**: 654–666, 1998.

[T.F.Smith81] T.F. Smith, M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, **147**: 195–197, 1981.

[V.A.Smith02] V.A. Smith, E.D. Jarvis, A.J. Hartemink. Evaluating functional network inference using simulations of complex biological systems. *Bioinformatics*, **18**: 216–224, 2002.

[V.A.Smith03] V.A. Smith, E.D. Jarvis, A.J. Hartemink. Influence of network topology and data collection on network inference. *Pacific Symposium on Biocomputing*, **8**: 164–175, 2003.

[Someren00] E.P. van Someren, L.F.A. Wessels, M.J.T. Reinders. Linear modeling of genetic networks from experimental data. *International Conference on Intelligent Systems for Molecular Biology*, **8**: 355–366, 2000.

[Someren02] E.P. van Someren, L.F.A. Wessels, E. Backer, M.J.T. Reinders. Genetic network modeling. *Pharmacogenomics*, **3**: 507–525, 2002.

[Somogyi96] R. Somogyi, C.A. Sniegoski. Modeling the complexity of genetic networks: understanding multigenic and pleiotropic regulation. *Complexity*, **1**: 45–63, 1996.

[Sonenshein01] A.L. Sonenshein, J.A. Hoch, R. Losick. *Bacillus subtilis and its Closest Relatives: From Genes to Cells*. American Society for Microbiology, Washington, D.C., 2001.

[Spellman98] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, **9**: 3273–3297, 1998.

[Szabo96] A. Szabo, R. Korszun, F.U. Hartl, J. Flanagan. A zinc finger-like domain of the molecular chaperone DnaJ is involved in binding to denatured protein substrates. *The EMBO Journal*, **15**: 408–417, 1996.

[Szallasi98] Z. Szallasi, S. Liang. Modeling the normal and neoplastic cell cycle with "realistic Boolean genetic networks": their application for understanding carcinogenesis and assessing therapeutic strategies. *Pacific Symposium on Biocomputing*, **3**: 66–76, 1998.

[Tamada03] Y. Tamada, S. Kim, H. Bannai, S. Imoto, K. Tashiro, S. Kuhara, S. Miyano. Estimating gene networks from gene expression data by combining Bayesian network model with promoter element detection. *Bioinformatics*, **19**: 227–236, 2003.

[Thieffry98] D. Thieffry, A.M. Huerta, E. Pérez-Rueda, J.C. Vides. From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in *Escherichia coli*. *BioEssays*, **20**: 433–440, 1998.

[Tinerey86] L. Tinerey, J.B. Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, **81**: 82–86, 1986.

[Valbuzzi01] A. Valbuzzi, C. Yanofsky. Inhibition of the *B. subtilis* regulatory protein TRAP by the TRAP-inhibitory protein, AT. *Science*, **293**: 2057–2059, 2001.

[Viterbi67] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, **13**: 260–269, 1967.

[Wagner01] A. Wagner. How to reconstruct a large genetic network from n gene perturbations in fewer than n^2 easy steps. *Bioinformatics*, **17**: 1183–1197, 2001.

[Watson53] J. Watson, F. Crick. A structure for deoxyribose nucleic acid. *Nature*, **171**: 737–738, 1953.

[Zhu00] G. Zhu, P.T. Spellman, T. Volpe, P.O. Brown, D. Botstein, T.N. Davis, B. Futcher. Two yeast forkhead genes regulate the cell cycle and pseudohyphal growth. *Nature*, **406**: 90–94, 2000.

Expression of Thanks

I am indebted to my supervisor, Professor Satoru Miyano, for guidance and for persistently encouraging me during my studies. I would also like to thank the members of the thesis committee, Professor Kenta Nakai, Professor Masami Hagiya, Professor Hiroshi Imai, Professor Jun'ichi Tsujii and Professor Akinori Yonezawa, for their helpful comments and suggestions. Advice from Hideo Bannai and Yoshinori Tamada was very helpful, especially for parallelising the gene network estimation software. I am also grateful for valuable discussions and advice from Seiya Imoto and Michiel de Hoon.
Finally, I would like to thank my wife for her loving support and understanding.