FROM GENE EXPRESSION TO MOLECULAR PATHWAYS

THESIS SUBMITTED FOR THE DEGREE OF "DOCTOR OF PHILOSOPHY"

BY

Dana Pe'er

SUBMITTED TO THE SENATE OF THE HEBREW UNIVERSITY

NOVEMBER 2003

This work was carried out under the supervision of

Nir Friedman

Abstract

Molecular networks involving interacting proteins, RNA, and DNA molecules, underlie the major functions of living cells. DNA microarrays probe how the gene expression changes to perform complex coordinated tasks in adaptation to a changing environment at a genome-wide scale. In this dissertation we address the challenge of reconstructing molecular pathways and gene regulation from gene expression data. Our goal is to automatically infer regulatory relations between genes, as well as other types of molecular interactions. To answer this challenge, we develop probabilistic graphical models of the biological system. We offer three such models and algorithms to automatically learn these from gene expression data. Our models and learning algorithms are based on the assumption that statistical correlation might indicate molecular or genetic interaction. We offer systematic evaluation for each of the methods presented culminating in experimental validation of novel predictions, automatically generated by one of our models.

Acknowledgements

I would like to express my deepest gratitude to my mentor, Nir Friedman. Nir is a genuine role model, and while I have had many teachers, Nir's mark is the most profound. Nir initiated me into the discipline of machine learning in graphical models and continuously taught me the most important scientific skills: how to dive deep into messy data and surface with simple models that address the question at hand, always striving to understand the connection between data, model and reality. Few have these skills and I was privileged to learn from a true master, I leave Nir with much yet to learn. Nir's contribution to the research in this thesis is fundamental, from the basic idea of coupling Bayesian networks with gene networks to little comments that made my presentation so much clearer.

I have spent a total of ten terrific years as a student at the Hebrew University and this has been a significant chapter in my life. During this time, many teachers have molded me into the researcher I am today. I would like to thank Avi Wigderson for patiently teaching me the rigors of problem solving. Avi is a mental giant and I was most privileged to brainstorm with him and learn how he takes a hard problem apart into little bits he can understand. I would like to thank Shmuel Peleg for teaching me that resarch should first and foremost be fun. Shmuel taught me that if one does not find enjoyment and passion in the problem at hand, it is probably the wrong problem to be working on. I rarely left his office without a smile. I was especially fortunate to an adopting "mother" and "father", Daphna Weinshall and Noam Nisan, in the Computer Science department. While never my official mentors, they took me under their wing, providing guidance, many rewarding discussions and emotional support. In addition, I would like to thank Noam for bringing to my attention α-modular functions and their connection to the MinReg algorithm. I would like to thank Daphna for actively fighting to make my years at the university more comfortable, be it easing the prerequisites when I transferred from mathematics or easing my TA workload as a new mother.

Good science is always the joint effort of many people and the research in this thesis is no exception. This thesis could never have happened without Aviv Regev, my scientific partner, biology tutor and dearest friend. My research is the result of a close and synergistic collaboration with Aviv, working with whom is an absolute joy and pleasure. Aviv transformed me from a naïve computer scientist to semi-biologist, teaching me so much more than biology along the way. In addition to sharing her wisdom and many unique insights, Aviv gave me endless support and backing. During the toughest and lowest points, Aviv was always there to stop me from quitting, by infecting me with her energetic enthusiasm and leading me to believe in myself. There are simply no words to express my gratitude to her.

I would like to give many thanks to all my co-authors on the works presented here. Michal Linial, the first biologist who dared believe our ideas might have merit. Iftach Nachman, who shared with me the first steps of this research. Gal Elidan, who brought order and efficiency to the chaos in which I was used to be working in. It was a wonderful pleasure to work with Amos Tanay on our 'underground' MinReg project, and the speed in which he programmed some of our ideas never ceased to surprise me. Amos has great scientific vision and I cherish the many hours we spent

brainstorming over coffee.

My intense collaboration with Eran Segal has been very fruitful and lead to great science. Eran, I very much admire your ability and stamina. I feel very priviliged to have worked with Daphne Koller, a brilliant scientist; I learned much from our many insightful discussions.

Lots of thanks to all my lab mates at the Computational Biology group and the Machine Learning group at the Hebrew university. It was marvelous to belong to a group with such great academic cooperation and social atmosphere; Full of seminars, reading groups, or just hallway discussions; Beach parties, dinners, and hiking trips. Specifically, I would like to thank my office mate Matan Ninio, who fed me well, almost as often as he distracted me. Matan was always helpful from the countless times he aided me with system related issues, to the laborious work of printing this thesis and submitting it for me.

I would also like to thank the many people who gave me the support and technical backing so I could focus on my research. I thank the Ministry of Science, Israel, for the Eshkol fellowship awarded to me and the Higher Education Council, Israel, for additional financing. I thank the System group at the Computer Science Department for the consistently providing the best and most reliable computer support possible. I thank the administrative staff at the Computer Science department for all their help and support, shielding me from the bureaucratic jungle that laid beyond our department. I would also like to thank Laura Garwin. Some times help comes unexpectedly, when my laptop crashed at critical stages of writing this thesis, Laura (at the time a stranger) out of pure kindness and generosity, lent me her personal laptop and hosted me in a wonderful office at the Bauer Center for Genomic Research.

During the course of my PhD. studies, the two most important events of my life occurred, the births of Inbar and Carmel. I would like to thank my two most beloved daughters for distracting me and granting me joy and happiness of a magnitude I never knew before. I apologize to them, it is Inbar and Carmel that have paid the heaviest price for this thesis, during the endless hours I worked away from them. I hope you understand and forgive. I thank Rocha, my mother-in-law for the countless hours she took care of the girls, giving me more time to work. While Rocha was with my girls, I could peacefully work, knowing they were getting the best of love and care. I thank my brother Michael for caring so much and for his constant reminders that there is so much more to life than research. I would like to thank Bat-Sheva for being available at any hour of the day or night for a relaxing walk and an opportunity to wind down.

I am grateful to both my parents, Mara and Aaron, for being such wonderful and supportive parents. They nurtured my curiosity, creativity and passion for understanding from the earliest age. I started my studies in the Mathematics department at the Hebrew university where both my parents met and the completion of this thesis gives me a great feeling of fulfillment. Dad, thank you for attempting to teach me Cantor's diagonal proof from preschool (that was a wee bit early), carefully correcting the English for this entire thesis and everything in between. Mom, you have been and will always be my role model, you are my very inspiration to excel, I aspire to be like you.

Last, I dedicate this thesis to my better half, Itsik. I am endlessly indebted and grateful to

Itsik for everything. My love, thank you for helping me in all aspects of my research. I thoroughly enjoyed our scientific discussions that occurred at all times of day and in all forms of dress. Many of your comments have been invaluable to my work. Thank you for your help with all my manuscripts including this one. Thank you for your unconditional love in my worse moments and for being a strong pillar of support in my most desperate moments. Thank you for making my victories more memorable by sharing them with you, this victory could have never happened without all your encouragement and help.

Contents

Chapter 1

Introduction

Molecular networks involving interacting proteins, RNA, and DNA molecules, underlie the major functions of living cells. Different metabolic, signaling and transcriptional levels are integrated to maintain a working cell. Deciphering the organization of molecular networks, their function and behavior under different conditions is a major goal of molecular cell biology. The availability of complete genomic sequences, combined with robotics, computing and material sciences, has lead to the development of high-throughput assays that probe cells at a new, genome-wide, scale. For instance, DNA microarrays [55, 90] can measure the mRNA levels of an entire genome in a single experiment. A major promise of such high-throughput methods, is that they will enable us to reconstruct how tens of thousands of genes and proteins work together in interconnected networks to orchestrate the basic functions of life.

In this dissertation we address the challenge of reconstructing molecular pathways and gene regulation from gene expression data. Our goal is to automatically infer regulatory relations between genes, as well as other types of molecular interactions. To answer this challenge, we develop *probabilistic models* of the biological system. A model is a simplification of the underlying system that captures the primary phenomena we are interested in and explains how these lead to the observations we make through our assays. We focus on *probabilistic* models that use stochasticity to account for measurement noise, variability in the biological system, and aspects of the system that are not captured by the model. In this thesis we formulate a number of such models, develop algorithms to learn the structure of these models from data and provide a systematic biological analysis for our resulting models.

1.1 Biological Background

We begin with a brief overview of the basic concepts of molecular biology - the interested reader is referred to molecular biology textbooks [2] for more information. Cells are the fundamental working units of every living system. To a large extent, cells are made of *proteins*, which determine the shape and structure of the cell. In addition, other proteins serve as machines that perform many

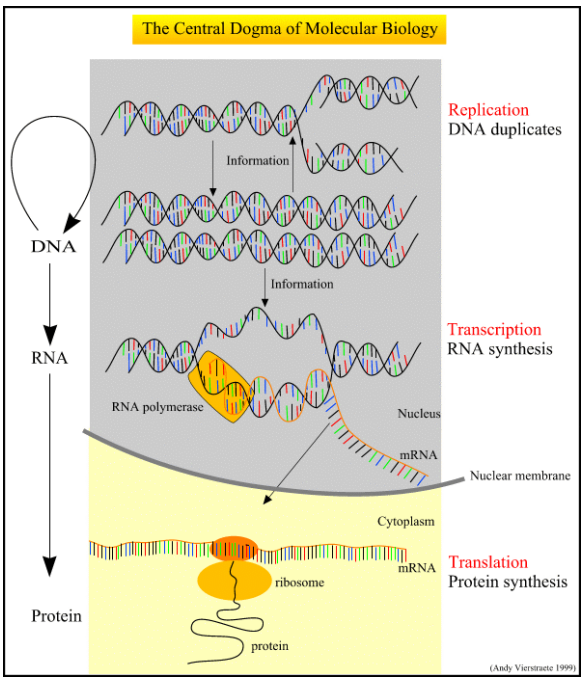

Figure 1.1: The central dogma of molecular biology

of life's functions, including molecular recognition and catalysis.

DNA is the organism's blueprint, it contains the instructions for the synthesis and regulation of proteins. Instructions for a particular protein is coded on a segment of DNA called a *gene*. The *central dogma* of molecular biology states that information flows from DNA through RNA to protein (see Figure 1.1). Thus, protein is synthesized from DNA in the following two step process:

1. DNA $\rightarrow$ RNA: *Transcription* is the process by which RNA polymerase copies a gene unto *mRNA* (messenger RNA) sequence using the DNA sequence as a template. This process by which a genes are transcribed into mRNA, present and operating in the cell, is termed *gene expression*.
2. RNA $\rightarrow$ Protein: In the subsequent process, called *translation*, a protein factory call ribosome, synthesizes the protein according the information coded in the mRNA.

A key observation is while each cell contains the same copy of the organism's DNA, the gene expression(and subsequently protein expression) can drastically vary, both temporally and spatially. To control gene expression, specialized proteins called *transcription factors* bind to the DNA and either enhance or inhibit the transcription of specific genes. These transcription factors often work together in different combinations, to ensure the correct amount of each gene is being transcribed. We note that transcription factions are themselves proteins and are thus subject to transcriptional control.

Transcription factors are by no means the only control over gene expression. Biological regulation is extremely diverse and involves different mechanisms at many layers: Before transcription

occurs, proteins regulate the structure of the DNA itself and determine whether a transcription factor can bind to the gene specific regulatory sites or not. Once the mRNA molecule is transcribed, other mechanisms regulate its editing and transport to the ribosome, thus controlling whether it gets translated into protein or not. For a given gene, the total amount of mRNA is regulated not only by transcription (creation) of mRNA, but also regulated by the degradation of mRNA. Regulation continues even after the protein is translated: a large part of biological regulation is via post-translational modifications that determine a protein's activity.

1.2 Microarrays

In recent years, technical breakthroughs in spotting hybridization probes and advances in genome sequencing lead to development of *DNA microarrays*, which consist of many species of probes, either oligonucleotides or cDNA, that are immobilized in a predefined organization to a solid surface. By using DNA microarrays researchers are now able to measure the abundance of thousands of mRNA targets simultaneously [26, 64], providing a "genomic" viewpoint of gene expression.

Microarray technology is based on *DNA hybridization*: a process in which a DNA strand binds to its unique complementary strand. A set of probes (known sequence) are fixed to a surface and are placed in interaction with a set of fluorescently tagged targets (unknown sequences). After hybridization, the fluorescently lit spots indicate the identity of the targets and the intensity of the fluorescence signal is in correlation to the quantitative amount of each target. Due to different hybridization affinities between clones and the fact that an unknown amount of cDNA is fixed for each probe, we cannot directly associate the hybridization level with a quantitative amount of transcript. Instead cDNA microarray experiments compare a reference pool and a target pool. Typically, green is used to label the reference pool, representing the baseline level of expression and red is used to label the target sample in which the cells were treated with some condition of interest. We hybridize the mixture of reference and target pools and read a green signal in case our condition reduces expression level and a red signal in case our condition increases expression level (see Figure 1.2).

A genome wide measurement of transcription is called an *expression profile* and provides us with a complete list of genes whose transcription level is effected in our condition. Biologically speaking, what we measure is how the *gene expression* of each gene changes to perform complex coordinated tasks in adaptation to a changing environment. In our context, while transcriptional regulation directly changes the measured mRNA levels, other factors such as proteins and their activity, are not observed by microarrays. Furthermore, due to biological variation and a multi-step experimental protocol, these data are very noisy, and fluctuate up to two-fold between repeated experiments.

In order to obtain a wide variety of profiles, reflecting different active pathways, various perturbations (e.g. mutations [51]) and treatments (e.g. heat shock [40]) are employed. The outcome is a matrix associating for each gene (row) and condition (column), the expression level. In our

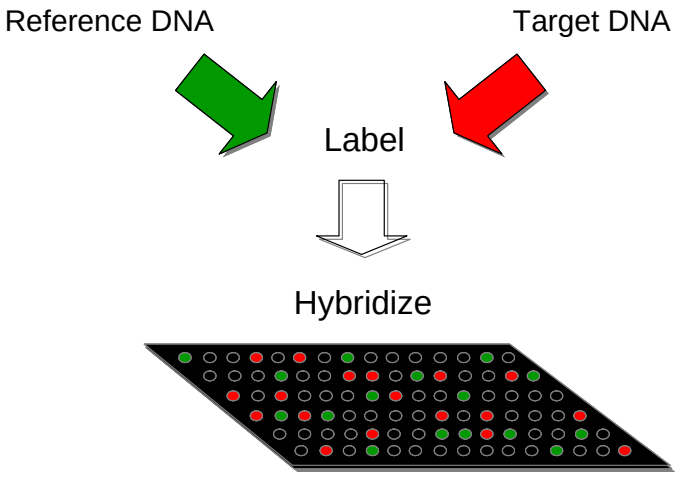

Figure 1.2: An image of a microarray. Each spot represents a different gene.

setting, this expression matrix contains thousands of gene and hundreds of conditions. Our goal is to uncover molecular interactions, most notably regulation, from these data.

1.3 Previous work

The first attempts to analyze these data identified a list of differentially expressed genes for each condition or treatment. Since current technology is very noisy and typical datasets contain only 2-5 repeats of each condition, even this simple task is not trivial. Early works [49] defined differential expression as a two-fold or greater change in expression. Developing statistically robust tests to determine which genes are differentially expressed remains an active area of research.

Currently, the most popular analysis method is *clustering*. Clustering of the genes is used to identify sets of genes that behave similarly (i.e. have similar expression patterns) over a set of experiments [3, 30] (see Figure 1.3). Clustering provides an intuitive way to organize and visualize of the data. Furthermore, clustering facilitates in the functional annotation of uncharacterized genes. If an uncharacterized gene belongs a cluster dominated by genes of some function, the unknown gene could possibly have a similar function. While clustering has successfully expanded our understanding in important biological processes (including cell cycle [30], cancer [3], metabolism [51]), it does not address our challenge to uncover the underlying gene network of interactions.

Previously, a number of regulatory models have been suggested. The most realistic of such models are stochastic networks [68]. While these directly model many of the actual details of the regulatory machinery, they are extremely complex and can only deal with small scale networks. For more global applications, simplified and abstract models are required. A few such models have been suggested, all based on the following basic idea: The regulatory network is a directed graph $\mathcal{G}$. Each node in $\mathcal{G}$ corresponds to a specific gene that behaves according to some deterministic function of its parents in $\mathcal{G}$. These include: Boolean network models [89, 1], where each gene is either on or

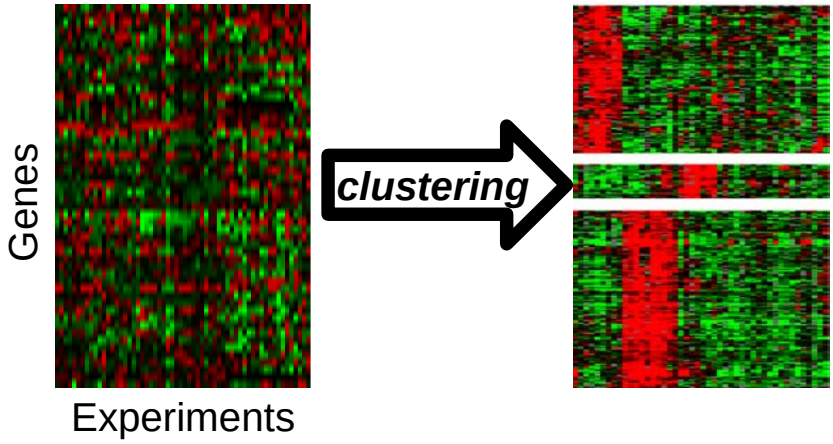

Figure 1.3: Clustering gene expression data: Each row corresponds to a gene and each column corresponds to a microarray sample, i.e., all the spots on the microarray in Figure 1.2 appear as a column in this figure. To the left is the unclustered input matrix. To the right is the matrix after clustering reordered the rows and columns.

off depending on some boolean function of its parents. Linear models [99, 27], where each gene is modeled as a continuous linear function of its parents. In order to simplify the complexity of such models, it is typically assumed that $\mathcal{G}$ is acyclic and of bounded indegree. While these methods have had partial success on simulated data, none of them have had any success when applied to real biological data.

1.4 Our approach

Recall, our goal is to reconstruct molecular networks representing processes such as gene regulation. To answer this challenge, we adopt a *systems* perspective of the cell and its components, and attempt to build models of this system. Our measurements observe the system at different *states*, which can be defined in terms of the concentration of active proteins and metabolites in the various compartments, the concentration of different mRNA molecules in the cytoplasm, etc. Our basic assumption is that the components in the cell do not work in isolation. Rather they effect each other through a wide variety of interactions. The key point being, that the components effect each other in a consistent fashion, Thus, if we consider a random sampling of the system, some states are more probable than others. For example, Gal4 is a transcription factor which strongly activates the galactose pathway genes, therefore if Gal4 is overexpressed in some state, it is likely that other galactose pathway genes are also overexpressed.

We treat measurements of the cell's components (e.g. gene expression measurements) as *random variables* and thus the likelihood of a cell state can be specified by the joint probability distribution on these variables. By representing measurements as random variables, to account for

measurement noise, variability in the biological system, and aspects of the system that are not captured by the model.

In this dissertation, due to issues of data availability, we only observe the level of mRNA expression for each of the genes. Therefore, we resort to a partial view which projects the activity of the entire cell onto gene expression profiles. In our model, each gene is associated with a random variable that represents the measurement of its expression. We use the term *genes*, interchangeably, to represent both the biological genes and the random variables that represent them in our model. We stress that the basic approach described here for gene expression data can be easily extended to other data types (e.g. protein levels) as these become available. For example, when more direct measurements of transcription factor activity become available, these be easily incorporated as random variables in the model and can greatly enhance the resulting reconstruction.

Our goal is to estimate the joint probability distribution over gene expression and understand its structural features from data. Our reconstruction of pathway structure is based on the following idea: molecular interactions between the genes sometimes generate corresponding statistical dependencies between the random variables that represent them. Using Gal4 as an example: Gal4 activates the transcription of other galactose genes, thus creating a correlation in their expression.

The learning algorithms presented in this dissertation detect consistent statistical dependencies and reconstruct a model that *explains* them, i.e., a model that could have generated the observed data. Our approach is global: we fit a model to data by studying the joint probability distribution over the entire gene set. Once we define such a model, its interpretation is as important an issue as the learning algorithm. An important question that will be repeatedly addressed throughout the dissertation is: What type of molecular relations create statistical dependencies in gene expression profiles?

A large part of this dissertation focuses on regulatory relations. Our ability to detect regulatory relations relies on the assumption that the gene expression profile of the regulators provides evidence as to their activity level. In order to capture a regulation event in gene expression data, we must observe concordant changes in the expression of both the regulator and its targets. While this is not always the case, we shall demonstrate that in many cases, this approach is capable of automatically reconstructing regulatory relations from gene expression profiles.

1.5 Road Map

In this dissertation, we present three different methods that infer molecular interactions and regulation from gene expression data. All three methods rely on the same assumption that statistical correlation might indicate molecular or genetic interaction. While these methods were designed for the the gene expression domain, each method is based on novel learning techniques that are applicable to other domains (e.g. documents or stock data). Evaluating these methods is not a trivial task: Since gene network reconstruction is a newly emerging field, there are no established benchmarks, nor known ground truth to which to compare to. Therefore, we emphasize the task of devising

systematic evaluations for our methods.

Chapter 2 is a tutorial on the technical background for this dissertation, providing both definitions and algorithms for the task of learning Bayesian networks. An intuitive description of Bayesian networks and how they can be used to model biology is presented in Section 3.2. Chapter 3 lays the foundations of our approach and describes how Bayesian networks can be tailored to analyze gene expression data. The method described infers detailed interactions of all types including transcriptional, signalling and metabolic. Parts of this chapter were published in Friedman et. al. [36] and Pe'er et.al. [76]. Chapter 4 provides details of the technical developments that solve two practical difficulties that arise when adapting Bayesian networks to the gene expression domain. Chapter 5 describes a specialized Bayesian network and learning algoirhm designed to focus the learning on regulatory relations between genes. Namely, using gene expression data, we demonstrate how to reconstruct regulatory relations on a *global* scale. This chapter is based on Pe'er et.al. [77]. Chapter 6 proposes a unified probabilistic framework that combines the best of both clustering and network modeling. This framework identifies modules of co-regulated genes, their regulators, and the shared regulation program that governs their behavior. These methods culminate in experimental validation of our method's predictions. This chapter is based on Segal et.al. [82, 81].

Chapter 2

Bayesian Networks Primer

Our goal is to model biological gene interactions derived from expression data. We believe that it is essential for such a model to be of a probabilistic nature for the following reasons. First, gene expression measurements are inherently noisy, and thus a probabilistic approach is required for robust analysis. Second, molecular biology is in itself a stochastic process. Finally, gene expression measurements provide only a partial picture of the state of the cell; many important parameters (e.g. phosphorylation state of a regulating protein) are, unfortunately, not revealed by gene expression measurements. A probabilistic model can handle uncertainty in such unobserved events. This thesis will present a number of probabilistic models for molecular networks based on the language and formalism of Bayesian networks [73]. This chapter will present the basic foundations of this formalism.

Bayesian networks provide a compact graphical representation of the joint probability distribution over $\mathcal{X} = (X_1, \ldots, X_n)$. Even for binary-valued variables, the joint distribution requires specification of the probabilities for the 2^n different assignments to $X_1 \ldots \mathcal{X}_n$. The key property of Bayesian networks is that they permit an explicit encoding of conditional independencies in a natural manner. These independencies can be used to represent such high dimensional data in a compact manner. Furthermore, these independencies are modeled using a qualitative graph structure that might correspond to structural aspects in the domain of interest. Therefore, the structural relations in the Bayesian network can be used to infer interactions between the different variables in the domain of interest.

In this chapter we will provide a brief overview of the formalism of Bayesian networks and the algorithms to learn such models from observed data. We begin with a number of notations. Consider a finite set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of random variables where each variable X_i may take on a value x_i from the domain $Val(X_i)$. In this thesis, we use capital letters, such as X, Y, Z, for variable names and lowercase letters x, y, z to denote specific values taken by those variables. Sets of variables are denoted by boldface capital letters $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, and assignments of values to the variables in these sets are denoted by boldface lowercase letters $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

At the core of Bayesian networks is the notion of *conditional independence*. We explain this

concept using the following example from classical genetics. Assume we are studying a certain mutation that appears in the Springfield population at a frequency of 0.001. If we sample a random resident of the city, we have Pr(Bart Simpson has mutation)$= 0.001$. Now, assume that we know that Bart's "Grandpa" (on his mothers side) has the mutation. In this case, Pr(Bart has mutation given Grandpa has mutation)$= 0.25$. Grandpa's genotype was informative towards Bart's genotype, the two genotypes are clearly dependent. Now we learn the additional information that Marge (Bart's mother) has the mutation too. In this case, Pr(Bart has mutation given Marge and Grandpa have mutation) = Pr(Bart has mutation given Marge has mutation but Grandpa does not) $= 0.5$. Likewise, if Marge does not have the mutation, the probability Bart has it is 0.001 regardless of whether Grandpa has the mutation or not. Once conditioned on Marge's genotype, Grandpa's genotype does not affect the probability of Bart's genotype. In our terminology we say that Bart's genotype is *conditionally independent* of Grandpa's genotype given Marge's genotype.

Definition 2.0.1: We say that $\mathbf{X}$ is *conditionally independent* of $\mathbf{Y}$ given $\mathbf{Z}$ if

$$P(\mathbf{X}|\mathbf{Y},\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})$$

and we denote this statement by $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$. ∎

2.1 Model Semantics

A *Bayesian network* is a structured graph representation of relationships between variables. The nodes represent the random variables in our domain and the edges often represent direct influence of one variable on another. More importantly, the graph represents conditional independencies between these variables. We now formally define Bayesian networks.

Definition 2.1.1: [73] A *Bayesian network* is a representation of a joint probability distribution consisting of two components. The first component, $\mathcal{G}$, is a *directed acyclic graph* (DAG) whose vertices correspond to the random variables $X_1, \ldots, X_n$. Let $\mathbf{Pa}_{X_i}$ denote the parents of X_i in $\mathcal{G}$. The second component, θ, describes a conditional probability distribution (CPD), $P(X_i|\mathbf{Pa}_{X_i})$, for each variable X_i in $\mathcal{X}$. ∎

Definition 2.1.2: The graph G encodes the *Markov Assumptions*: Each variable X_i is independent of its non-descendants, given its parents in $\mathcal{G}$.

$$\forall X_i (X_i \perp \textit{NonDescendants}_{X_i} \mid \mathbf{Pa}_{X_i})$$

∎

As an example we will see how a Bayesian network can represent the relations between 5 different genes. Assume that A is a transcription factor that binds to genes B and D, B is a transcription

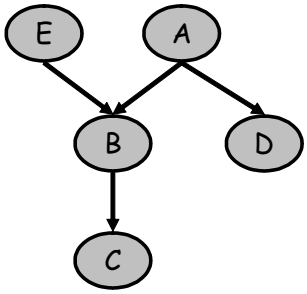

Figure 2.1: An example of a simple Bayesian network structure. This network structure implies several conditional independence statements:$(A \perp E)$,$(B \perp D \mid A, E)$, $(C \perp A, D, E \mid B)$, $(D \perp B, C, E \mid A)$, and $(E \perp A, D)$. The joint distribution has the product form $P(A, B, C, D, E) = P(A)P(E)P(B|A, E)P(C|B)P(D|A)$

factor that binds to C and E is a transcription factor that binds to B. Figure 2.1 shows an example of a Bayesian network structure $\mathcal{G}$ that captures these relations. The figure also lists the Markov independencies it encodes, and the product form (see below) they imply. We can see how the different relations between the genes are encoded in the graph structure. For instance, B is a function of A and E, its two regulating transcription factors. The genes B and D are co-regulated and thus their expression is dependant, but conditioned on their common regulator A, they become independent.

The two components, $\mathcal{G}$ and θ, specify a unique distribution on $X_1, \ldots, X_n$. By applying the chain rule of probabilities and properties of conditional independencies, any joint distribution that satisfies the conditional independence in definition Definition 2.1.2 can be decomposed into the product form

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}_{X_i}) \tag{2.1}$$

This is called the *chain rule for Bayesian networks*. This product form makes a Bayesian network representation of a joint probability compact and economizes the number of parameters. As an example, consider the joint probability distribution $P(A, E, B, C, D)$ represented in Figure 2.1. By the chain rule of probability, without any independence assumptions: $P(A, E, B, C, D) = P(A)P(E|A)P(B|A, E)P(C|A, E, B)P(D|A, E, B, C)$. Assuming all variables are binary, this representation requires $1 + 2 + 4 + 8 + 16 = 31$ parameters. Taking the conditional independencies into account $P(A, E, B, C, D) = P(A)P(E)P(B|A, E)P(C|B)P(D|A)$, which only requires $1 + 1 + 4 + 2 + 2 = 10$ parameters. More generally, given n binary variables and $\mathcal{G}$ whose indegree (i.e. maximal number of parents) is bounded by k, then instead of representing the joint distribution with $2^n - 1$ independent parameters we can represent it with at most $2^k n$ independent parameters. The *conditional probability distribution (CPD)*, $P(X_i|\mathbf{Pa}_{X_i})$ can be viewed as a probabilistic function of X_i whose inputs are X_i's parents in $\mathcal{G}$. In fact, these two dual views of a Bayesian network are equivalent: any distribution P satisfying the conditional independencies in Definition 2.1.2 can be encoded as a Bayesian network with structure $\mathcal{G}$ and associated CPDs.

A graph $\mathcal{G}$ specifies a product form as in Eq. (2.1). To fully specify a joint distribution, we also need to specify the conditional distributions, $P(X_i|\mathbf{Pa}_{X_i})$ for each variable X_i. We denote the parameters that specify these distributions by θ. When specifying these conditional distributions, we can choose from almost any computable representation. In this thesis we focus on discrete variables, though many continuous CPDs have been used for Bayesian networks [36, 37, 42].

The most general representation is a conditional probability table (CPT). Each row in these tables corresponds to a specific joint assignment $\mathbf{pa}_{X_i}$ to $\mathbf{Pa}_{X_i}$, and specifies the probability vector for X_i conditioned on $\mathbf{pa}_{X_i}$. For example, if $\mathbf{Pa}_{X_i}$ consists of k binary valued variables, the table will specify 2^k distributions. This general representation can describe any discrete conditional distribution. This flexibility comes at a price: The number of free parameters is exponential in the number of parents.

As an example assume that A and E each weakly activate gene B and together they strongly activate gene B. Following is an example CPT that represents such a relation.

a	e	$P(b=0)$	$P(b=1)$
a^0	e^0	0.96	0.04
a^0	e^1	0.61	0.39
a^1	e^0	0.68	0.32
a^1	e^1	0.07	0.93

Now assume a similar but slightly different scenario. Assume again that A weakly activates B, A and E together strongly activate B, and without A, E has no affect on B. In this case, the CPT would look like:

a	e	$P(b=0)$	$P(b=1)$
a^0	e^0	0.96	0.04
a^0	e^1	0.96	0.04
a^1	e^0	0.68	0.32
a^1	e^1	0.07	0.93

Notice that the first two rows are redundant, they represent the exact same conditional probability over B for two different value assignments to $\mathbf{Pa}_B$. A natural representation that captures such behavior is a *CPD-tree*. This is a *Context specific* CPD, that uses value assignments for only a subset of $\mathbf{Pa}_{X_i}$, to derive $P(X_i|\mathbf{Pa}_{X_i})$.

Definition 2.1.3: A *CPD-tree* for the distribution $P(X_i|\mathbf{Pa}_{X_i})$ is a structured CPD that consists of two types of nodes: decision nodes and leaf nodes. Each leaf node represents a distribution for X. Each decision node is labeled with a query $Y = y?$ s.t. $Y \in \mathbf{Pa}_{X_i}$ and $y \in Val(Y)$. Given a set of parent assignments, we begin at the root and traverse down the tree in a *path* that depends on the answers to the queries along the path. $P(X_i|\mathbf{Pa}_{X_i})$ is the probability distribution at the leaf node reached by such a traversal using the values of $\mathbf{Pa}_{X_i}$. ∎

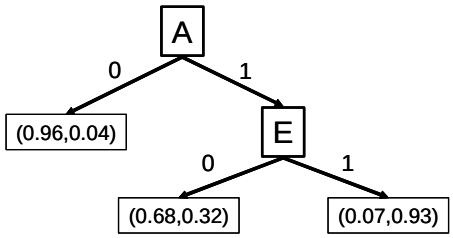

Figure 2.2: An example of a tree-CPD. Each square represents a query on a variable (A and E). Each edge is annotated with the value of its associated query node. The leaves contain probability distributions. Notice that when A is off, the value of E does not matter

Usually a path down the tree does not require an assignment to all variables in $\mathbf{Pa}_X$. This provides a compact, context specific representation of the CPD, which is suited for biological applications. If a given transcription factor only activates a target under specific conditions, its value is irrelevant unless those conditions are met. In our previous example, if E only works when A is active, we need to query its value only when $A = 1$. Figure 2.2 gives the associated CPD-tree for this example.

The tree representation has a number of advantages. It requires less parameters. The context specific relations can further provide biological understanding. In addition, many automated learning algorithms [9] exist to construct such trees from data. When the sample size is small, the fact this representation has less parameters is a crucial for the robustness of the learning.

2.2 The Graph structure: Independence, Dependence and Causality

In this section we discuss different aspects of the Bayesian network structure $\mathcal{G}$. We describe the relationship between the graph structure and conditional independencies the structure implies. First, we show how the graph structure can be used to make conditional independence queries. Second, we introduce the notion of equivalent structures. Finally, we describe the possible implications of these dependencies on the issue of causality.

2.2.1 d-separation

If we assume that genetic interactions can be modeled by probabilistic dependencies, then we can use independence queries on the distribution $P(X_1, \ldots, X_n)$ to infer these genetic interactions. The Bayesian network structure $\mathcal{G}$ greatly facilitates efficient multivariate independence queries. For instance, the structure $X \rightarrow Y \rightarrow Z$ implies that while X and Z are dependent, the variable Y renders them conditionally independent, $(X \perp Z \mid Y)$. The variable Y *explains away* the dependence between X and Z, hinting that the interaction between X and Z is indirect. Thus, a Bayesian network can be used to distinguish between direct and indirect dependencies. While this independence statement was easy to infer from the simple sub-structure ($X \rightarrow Y \rightarrow Z$), it

is natural to ask about queries which relate to variables who are further apart in $\mathcal{G}$. It is possible to automatically derive such conditional independence relations between such variables from the graph structure itself.

Before continuing, we present a graph sub-structure that plays a key role both for the notion of d-separation and for the notion of equivalent graphs.

Definition 2.2.1: [73] A *v-structure* is an induced sub-graph of the form $X \rightarrow Y \leftarrow Z$ so that no edge exists between X and Z. ∎

The v-structure implies an interesting set of dependencies. In the previous cases, X and Z were dependent only when Y was unobserved; in a v-structure, given the value of Y, two possibly independent variables become dependant. A classic example of such a dependency is from genetics: Consider random variable Y representing the existence of a rare mutation in some child and and X, Z representing the existence of the same mutation in each of that child's two biological parents. The genotype of each of the parents X and Z is independent of one another. But, if we know that $Y = 1$, i.e. the child has the rare mutation, this means one of the two parents must have the mutation. Now if we are also given that $X = 0$ we can infer that $P(Z = 1|Y = 1, X = 0) = 1$ and given $X = 1$ we can infer that $P(Z = 1|Y = 1, X = 1) = \epsilon$. Therefore given the value of Y, the independent variables X and Z become dependent.

Intuitively, one can view dependence as a property that can "flow" between the nodes representing X and Y through paths that connect them in the $\mathcal{G}$. For instance, take the example $X \rightarrow Y \rightarrow Z$, dependence can "flow" from X to Z through Y, unless Y "blocks" this flow. Since $(X \perp Z \mid Y)$, the path is "blocked" only when Y is given. In an opposite case $X \rightarrow Y \leftarrow Z$, dependence can "flow" from X to Z only if Y is given. We generalize these notions to longer paths. We say the graph has a trail from X_1 to X_n, denoted $X_1 - \ldots - X_n$. If for every $i = 1..n-1$, $\mathcal{G}$ contains either $X_i \rightarrow X_{i+1}$ or $X_i \leftarrow X_{i+1}$.

Definition 2.2.2: Let $\mathcal{G}$ be a Baysian network structure and $X_1 - \ldots - X_n$ be a trail in $\mathcal{G}$. Let $\mathbf{E} \subset \mathbf{X}$ be a subset of nodes. We say the there is an active trail between X_1 and X_n given evidence $\mathbf{E}$ if:

- Whenever we have a v-structure $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ then X_i or one of its descendants are in $\mathbf{E}$.
- No other node along the trail is in $\mathbf{E}$.

∎

Intuitively this means that the dependence can "flow" through every triplet $X_{i-1} - X_i - X_{i+1}$.

Definition 2.2.3: [43] Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be three sets of nodes in $\mathcal{G}$. We say that $\mathbf{X}$ and $\mathbf{Y}$ are *d-separated* given evidence $\mathbf{Z}$, denoted *d-sep*$_{\mathcal{G}}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z})$, if there is no active trail between any node $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$ given evidence $\mathbf{Z}$. ∎

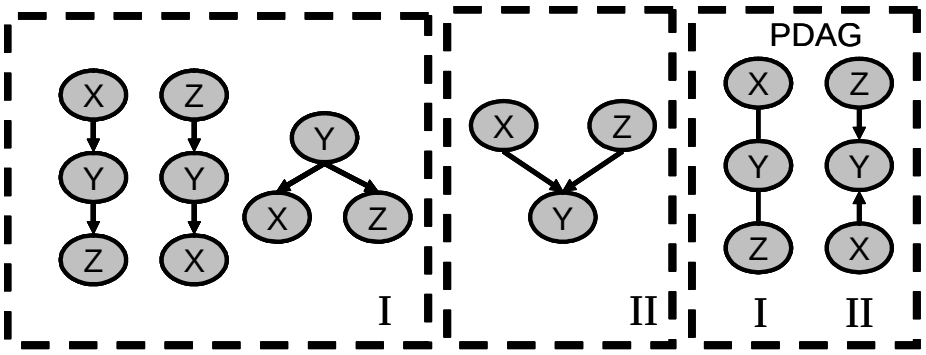

Figure 2.3: The skeleton $X - Y - Z$ is partitioned into two equivalence classes: I representing $((X \perp Y \mid Z), \neg(X \perp Z \mid \emptyset))$ and II the v-structure representing $\{\neg(X \perp Y \mid Z), (X \perp Z \mid \emptyset)\}$. The right pane illustrates the corresponding PDAGs.

Denote Ind($\mathcal{G}$) be the set of independence statements (of the form X is independent of Y given $\mathbf{Z}$) that are implied by $\mathcal{G}$. Using this formulation of *d-separation* we can check if any such conditional independence statement holds using a linear time graph algorithm [73].

2.2.2 Equivalence Classes

More than one graph can imply exactly the same set of independencies. For example, consider the graphs $X \rightarrow Y$ and $X \leftarrow Y$, both imply the same set of independencies (i.e., Ind($\mathcal{G}$) $= \emptyset$).

Definition 2.2.4: Two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are *equivalent* if Ind($\mathcal{G}_1$) $=$ Ind($\mathcal{G}_2$). That is, both graphs are alternative ways of describing the same set of independencies .∎

This notion of equivalence is crucial, since when we examine observations from a distribution, we cannot distinguish between equivalent graphs[1]. Pearl and Verma [75] show that we can characterize *equivalence classes* of graphs using a simple representation. In particular, these results establish that equivalent graphs have the same underlying undirected graph but might disagree on the direction of some of the arcs.

Theorem 2.2.5: [75] Two Bayesian network structures are equivalent if and only if they have the same underlying undirected graph (termed *skeleton*) and the same v-structures.

For example, the skeleton $X - Y - Z$ can be partitioned into two equivalence classes. One containing three graphs representing $((X \perp Y \mid Z), \neg(X \perp Z \mid \emptyset))$ and the v-structure representing $(\neg(X \perp Y \mid Z), (X \perp Z \mid \emptyset))$ (see Figure 2.3).

[1]To be more precise, under the common assumptions in learning networks, which we also make in this thesis, one cannot distinguish between equivalent graphs. If we make stronger assumptions, for example by restricting the form of the conditional probability distributions we can learn, to a tree, for example, we might have a preference of one equivalent network over another.

Moreover, an equivalence class of network structures can be uniquely represented by a *partially directed graph* (PDAG) $\mathcal{P}$, where a directed edge $X \rightarrow Y$ denotes that all members of the equivalence class contain the directed edge $X \rightarrow Y$; an undirected edge $X - Y$ denotes that some members of the class contain the directed edge $X \rightarrow Y$, while others contain the directed edge $Y \rightarrow X$.

Assume we are given some Bayesian network structure $\mathcal{G}$ and wish to derive the PDAG $\mathcal{P}$ representing $\mathcal{G}$'s equivalence class. Since Theorem 2.2.5 states that all equivalent graphs must agree on their v-structures, it is obvious that we need to orient any edge that participates in a v-structure. What is less obvious is that there are other edges in the graph that need to be oriented. These are the edges that form new v-structures when reversed. They can be derived through one of the following three propagation rules for *compelled edges* in $\mathcal{P}$ (see Figure 2.4):

- Consider the following subgraph $X \rightarrow Y - Z$, where no edge exists between X and Z. Each edge direction between Y and Z defines a different equivalence class. The edge $Y \leftarrow Z$ forms a v-structure, while $Y \rightarrow Z$ does not. Therefore the edge in the corresponding PDAG $\mathcal{P}$ is compelled to be directed as $Y \rightarrow Z$.
- Consider the following subgraph $X \rightarrow Y$, $Y \rightarrow Z$, and $X - Z$. If we direct the edge as $Z \rightarrow X$, a cycle is formed. Therefore, to ensure acyclicity, the edge is compelled to be directed as $X \rightarrow Z$.
- Consider the following subgraph $X - Y$, $X - W$, $X - Z$, $Y \rightarrow Z$, and $W \rightarrow Z$. The edge $X - Z$ is compelled to be directed as $X \rightarrow Z$. Assume that the edge is directed as $X \leftarrow Z$. Then to avoid acyclicity, the edges $Y \rightarrow X$ and $W \rightarrow X$ are compelled, thus forming a new v-structure.

Given a DAG $\mathcal{G}$, the PDAG representation of its equivalence class can be constructed as follows. We begin from the underlying skeleton of $\mathcal{G}$ and orient all edges which participate in a v-structure. Then we continue applying the propagation rules of Figure 2.4 until no more subgraphs corresponding to one of the rules exist.

Proposition 2.2.6: *If we apply the procedure described above to $\mathcal{G}$, the resulting PDAG represents the equivalence class of $\mathcal{G}$.*

It is interesting to characterize not only the set of reversible edges in a PDAG, but given a specific DAG, we wish to characterize the set of edges that can be reversed so that the resulting DAG remains equivalent. The following definition characterizes such edges:

Definition 2.2.7: An edge $X \rightarrow Y$ is *covered* in $\mathcal{G}$ if $\mathbf{Pa}_Y = \mathbf{Pa}_X \cup X$. ∎

The following clarifies the relation between covered edges and reversible edges: An edge is reversible if it does not participate in a v-structure and does not create a new v-structure when reversed. If $\mathbf{Pa}_X = \mathbf{Pa}_Y = \emptyset$, then the edge $X - Y$ obviously can not involve any v-structures and

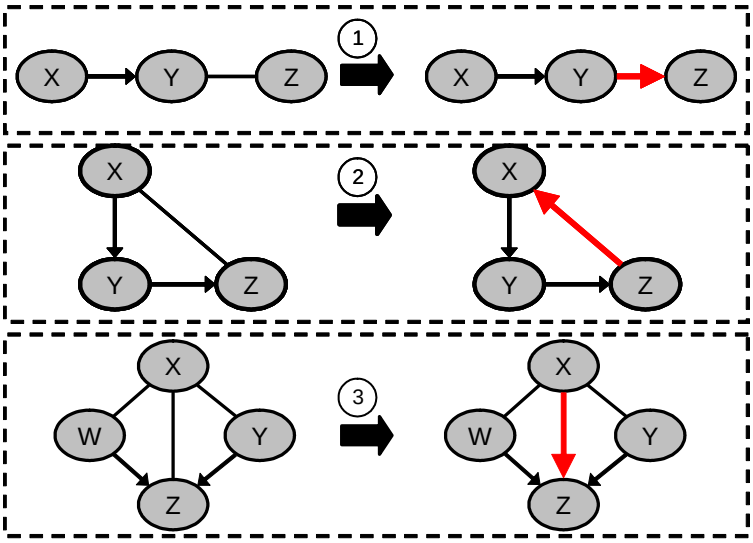

Figure 2.4: Propagation rules for compelled edges After all v-structures are oriented, by iteratively applying these rules to a PDAG, a representation of the equivalence class is constructed

can be reversed. For any $Z \in \mathbf{Pa}_Y$, the edges $X \rightarrow Y$ and $Z \rightarrow Y$ do not constitute a v-structure due to the edge $Z \rightarrow X$. If we reverse $X \rightarrow Y$ then $Y \rightarrow X$ and $Z \rightarrow X$ do not constitute a v-structure due to the edge $Z \rightarrow Y$. Notice that if $X \rightarrow Y$ is covered, then after the reversal the edge $Y \rightarrow X$ remains covered as well.

Using the concept of covered edges, Chickering [13] shows that one can transform any structure $\mathcal{G}_1$ to any other equivalent structure $\mathcal{G}_2$ via a series of single reversals of covered edges so that all intermediate graphs remain in the same equivalence class. This transformation greatly simplifies proving invariant properties of equivalent structures, since it is enough to prove the invariance between two graphs that only differ in the orientation of a single covered edge. Chickering [13] uses this characterization to derive an efficient algorithm to identify all compelled edges in a structure. For a graph with m edges and n nodes, the complexity of this algorithm is $O(m \log n)$.

Theorem 2.2.8: [13] Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be any pair of equivalent graphs. Let $\Delta(\mathcal{G}_1, \mathcal{G}_2)$ denote the set of edges in $\mathcal{G}_1$ that have an opposite orientation in $\mathcal{G}_2$. Then there exists a sequence of $|\Delta(\mathcal{G}_1, \mathcal{G}_2)|$ distinct edge reversals applied to $\mathcal{G}_1$ with the following properties:

- After each reversal, the resulting graph $\mathcal{G}$ is equivalent to $\mathcal{G}_1$.
- After all reversals, the resulting graph is identical to $\mathcal{G}_2$.
- Each edge reversed in $\mathcal{G}$ is an covered edge.

2.2.3 Causality

Recall that, a Bayesian network is a model of dependencies between multiple measurements. However, we are also interested in modeling the mechanisms that generated these dependencies. Thus, we want to model the flow of causality in the system of interest (e.g., gene transcription in our gene expression domain). A *causal network* is a model of such causal processes. Having a causal interpretation facilitates predicting the effect of an *intervention* in the domain: setting the value of a variable in such a way that the manipulation itself does not affect the other variables.

While at first glance there seems to be no direct connection between probability distributions and causality, causal interpretations for Bayesian Networks have been proposed [75, 74]. A causal network is mathematically represented similarly to a Bayesian network, a DAG where each node represents a random variable along with a local probability model for each node. However, causal networks have a stricter interpretation on the meaning of edges: the parents of a variable are its *immediate causes*.

A causal network models not only the distribution of the observations, but also the effects of *interventions*. If X causes Y, then manipulating the value of X affects the value of Y. On the other hand, if Y causes X, then manipulating X will not affect Y. Thus, although $X \rightarrow Y$ and $X \leftarrow Y$ are equivalent Bayesian networks, they are not equivalent causal networks.

A causal network can be interpreted as a Bayesian network when we are willing to make the *Causal Markov Assumption*: given the values of a variable's immediate causes, it is independent of its earlier causes. When the casual Markov assumption holds, the causal network satisfies the Markov independencies of the corresponding Bayesian network. For example, this assumption is a natural one in models of genetic pedigrees: once we know the genetic makeup of the individual's parents, the genetic makeup of her ancestors is not informative about her own genetic makeup.

The central issue is: When can we learn a causal network from observations? This issue received a thorough treatment in the literature [47, 75, 91]. We briefly review the results relevant for this thesis. For a more detailed treatment of the topic, we refer the reader to [74, 19].

First it is important to distinguish between an *observation*: a passive measurement of our domain (i.e., a sample from $\mathcal{X}$) and an *intervention*: setting the values of some variables using forces outside the causal model (e.g., gene knockout or over-expression). It is well known that interventions are an important tool for inferring causality. What is surprising is that occasionally, some causal relations can be inferred from observations alone.

To learn causality, several assumptions have been made. First, a modeling assumption: we assume that the (unknown) causal structure of the domain satisfies the Causal Markov Assumption. Thus, we assume that causal networks can provide a reasonable model of the domain. Some of the results in the literature require a stronger version of this assumption, namely that causal networks can provide a perfect description of the domain (that is an independence property holds in the domain if and only if it is implied by the model). The second assumption is that there are no *latent* or hidden variables that effect several of the observable variables. Unfortunately the second assumption does not hold in our domain, thus causal conclusions from our learning procedure must

be treated with caution.

If we do make these two assumptions, then we essentially assume that one of the possible DAGs over the domain variables is the "true" causal network. However, as discussed above, from observations alone, we cannot distinguish between causal networks that specify the same independence properties, i.e., belong to the same equivalence class (see section 2.2.2). Thus, at best we can hope to learn a description of the equivalence class that contains the true model. In other words, we will learn a PDAG description of this equivalence class.

Once we identify such a PDAG, we are still uncertain about the true causal structure in the domain. However, we can draw some causal conclusions. For example, if there is a directed path from X to Y in the PDAG, then X is a causal ancestor of Y in *all* the networks that could have generated this PDAG including the "true" causal model. Thus, in this situation we can recover some of the causal directions.

2.3 Learning Bayesian Networks

The learning task deals with the following situation: We are given a *training set* of samples $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$ that are independently drawn from some unknown generating Bayesian network $\mathcal{G}^*$ with an underlying distribution P^*. Our goal is to recover $\mathcal{G}^*$. Since $\mathcal{D}$ is only a small noisy sample from P^*, we can not detect, with complete reliability, which dependencies are present in the underlying distribution. Instead, we search for a relatively simple model, $\mathcal{B} = \langle \mathcal{G}, \theta \rangle$ (with few edges), that was likely to have generated the data, i.e., a $\mathcal{B}$ whose underlying distribution is close to the empirical distribution of the data $\mathcal{D}$.

More precisely, we search for an equivalence class of networks that best matches $\mathcal{D}$. Recall that all structures in an equivalence class represent the same dependencies and are equally close to the empirical distribution of $\mathcal{D}$. We can not distinguish between them based solely on the data $\mathcal{D}$. Thus the best we can hope for is to recover a structure that is equivalent to $\mathcal{G}^*$.

The theory of learning networks from data has been examined extensively over the last decade. In this thesis we take the *score based* approach to learning. We define a hypothesis space of potential network models, introduce a statistically motivated scoring function that evaluates each network with respect to the training data, and to search for the highest scoring network.

In this section we describe a statistically motivated score and how to find the corresponding high scoring network. First, we assume that the graph structure $\mathcal{G}$ is given and describe an appropriate score for the CPD parameters θ and a closed form solution for the highest scoring parameters. Then, we describe an appropriate score for the graph structure itself. Finally, we describe a greedy algorithmic approach that finds a high scoring network structure.

2.3.1 Parameter Estimation

Maximum Likelihood Estimation

In this section we assume that the structure of the graph $\mathcal{G}$ is known. While this is not a reasonable assumption for our domain, the theory of parameter estimation is a basic building block for the structure learning described in Section 2.3.2. First we define when a Bayesian network is *good*, i.e., one that *fits* the data $\mathcal{D}$? In the Bayesian network learning task we implicitly assume that there is some Bayesian network $\mathcal{B}^*$ that generated the data $\mathcal{D}$ and our goal is to use this data in order to try to reconstruct $\mathcal{B}^*$. Therefore, a good Bayesian network $\mathcal{B}$ is one that is likely to have generated $\mathcal{D}$. If we assume that $\mathcal{G}$ is already known, our task then is to find the conditional probabilities θ which maximize the likelihood of $\mathcal{D}$.

Definition 2.3.1: We define a *likelihood function*, $L(\theta : \mathcal{D})$, which measures the likelihood of the data.

$$L(\theta : \mathcal{D}) = \prod_{m=1}^{M} P(\mathbf{x}[m] \mid \theta)$$

∎

The idea behind *Maximum likelihood estimation* is given a dataset $\mathcal{D}$, we wish to choose parameters $\hat{\theta}$ that maximize the likelihood of the data:

$$\hat{\theta} = \max_{\theta} L(\theta : \mathcal{D}) \tag{2.2}$$

Eq. (2.2) could potentially be a hard expression to optimize, due to the high dimensionality of θ and the large number of parameters that need to be concurrently optimized. One of the big advantages of the Bayesian network representation is that this likelihood decomposes into local likelihood functions. Not only does this simplify the calculation of the likelihood, more importantly it renders finding its optimal parameters tractable. Each local likelihood can be optimized in an independent manner, thus decomposing a complex global problem into smaller sub-problems.

$$\begin{aligned} L(\theta : \mathcal{D}) &= \prod_{m=1}^{M} P(\mathbf{x}[m]) \\ &= \prod_{m=1}^{M} \prod_{i=1}^{n} P(x_i[m] \mid \mathbf{pa}_{X_i}[m] : \theta) \\ &= \prod_{i=1}^{n} \left[\prod_{m=1}^{M} P(x_i[m] \mid \mathbf{pa}_{X_i}[m] : \theta) \right] \\ &= \prod_{i=1}^{n} L_i(\theta_{X_i|\mathbf{pa}_{X_i}} : \mathcal{D}) \end{aligned}$$

where $L_i(\theta_{X_i|\mathbf{Pa}_{X_i}} : \mathcal{D}) = \prod_{m=1}^{M} P(x_i[m] \mid \mathbf{pa}_{X_i}[m] : \theta)$ is the *local likelihood function* for X_i.

In the case of our table CPDs this local likelihood can be further decomposed into simple tractable form. Suppose we have a variable X with its parents $\mathbf{Pa}_{X_i}$, then we have a parameter $\theta_{x|\mathbf{u}}$ for each combination of $x \in Val(X)$ and $\mathbf{u} \in Val(\mathbf{Pa}_{X_i})$. The idea behind the decomposition is to group together all the instances in which $X = x$ and $\mathbf{U} = \mathbf{u}$. We denote $M[x, \mathbf{u}]$ to be the number of instances in which $X = x$ and $\mathbf{U} = \mathbf{u}$ and $M[\mathbf{u}] = \sum_{x \in X} M[\mathbf{u}, x]$. Then by rearranging the order of the product we can write

$$L_i(\theta_{X|U} : \mathcal{D}) = \prod_{\mathbf{u} \in Val(\mathbf{Pa}_{X_i})} \prod_{x \in Val(X)} \theta_{x|\mathbf{u}}^{M[x,\mathbf{u}]} \tag{2.3}$$

Proposition 2.3.2: *The maximal likelihood estimation (MLE) for a Bayesian network with multinomial table CPDs is given by:*

$$\hat{\theta}_{x|\mathbf{u}} = \frac{M[x, \mathbf{u}]}{M[\mathbf{u}]} \tag{2.4}$$

We call the counts $M[x, \mathbf{u}]$ and $M[\mathbf{u}]$ *sufficient statistics*. Given these counts, that actual data instances $x[1] \ldots x[M]$ themselves are no longer needed. The sufficient statistics summarize all the relevant information from the data that is needed in order to calculate the likelihood. Notice that the optimal parameters are based on the empirical counts observed in our data. Thus optimizing the likelihood is equivalent to finding the best approximation for the empirical distribution constrained to the independencies of $\mathcal{G}$.

Bayesian Approach

While the MLE approach seems like a suitable score for measuring the fit of a Bayesian network to the data, it has a number of disadvantages. Its main drawback is that it tends to overfit the model to the particular data instance at hand. This problem is critical in our domain, in which the number of samples is relatively small. We illustrate this problem using an example from the medical domain: Assume we are performing a study on the effect of smoking on lung cancer. Our sample contains 30 non-smokers, none of which contracted lung cancer. MLE would construct a model that postulates P(lung cancer=YES | smoker=NO) = 0. We *believe* that the correct answer is that there is a small chance for a non-smoker to develop lung cancer, but our small sample did not contain such a case.

We therefore turn to the *Bayesian approach*, which formulates this concept of prior belief in a principled manner. The idea is that in addition to the observed data $\mathcal{D}$, we have some initial distribution, $P(\theta)$ termed the *prior*, which encodes our beliefs regarding the domain prior to our observations. When we have little prior knowledge of our domain this distribution is often flat and mostly ensures that every event has some non-zero probability. On the other hand, if we do have specific belief about our domain, this distribution can be more peaked over certain values.

After we observe some data $\mathcal{D}$ we update the distribution $P(\theta)$, to reflect the combination of both our prior belief and observations. This updated distribution, denoted $P(\theta \mid \mathcal{D})$, is called the

posterior distribution.

$$P(\theta \mid D) = \frac{P(\mathcal{D} \mid \theta)P(\theta)}{P(\mathcal{D})}. \tag{2.5}$$

The term $P(\mathcal{D})$, termed the *marginal likelihood*, averages the probability of the data over all possible parameter assignments. Since it is a normalizing constant, which is independent of θ, we ignore it in our score calculations.

Definition 2.3.3: The Gamma function $\Gamma(x)$ is defined to be an extension of the factorial to real number arguments. If n is a natural number, then $\Gamma(n) = (n-1)!$. The Gamma function is defined as the following integral:

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-x}dt \tag{2.6}$$

∎

In this thesis we use the *Dirichlet priors* [25] for multinomial distributions. A Dirichlet prior is specified by a set of *hyperparameters* $\alpha_{x^1|\mathbf{u}}, \ldots \alpha_{x^K|\mathbf{u}}$, one such hyperparameter corresponding to each $x^j \in Val(X)$. The Dirichlet distribution is specified by:

$$P(\theta) = Dirichlet(\alpha_{x^1|\mathbf{u}}, \ldots \alpha_{x^K|\mathbf{u}}) = \frac{\Gamma(\alpha_*)}{\prod_l \Gamma(\alpha_{x^l|u})} \prod_j \theta_{x^j|u}^{\alpha_{x^j|u}-1} \tag{2.7}$$

where Γ is the Gamma function and $\alpha_* = \sum_j \alpha_{x^j|u}$. One can view α_* as our *effective sample size*, that is we assume that our prior is based on α^* observations. This reflects how strongly we believe in our prior. As we accumulate more samples in $\mathcal{D}$, the effect of the prior on the posterior grows weaker.

Dirichlet priors have a number of desirable properties; they satisfy *global parameter independence* and *local parameter independence*. This means that the prior decomposes into a product of independent terms in a similar manner to the decomposition of MLE.

Definition 2.3.4: A parameter prior $P(\theta)$ is said to satisfy *global parameter independence* if it decomposes into the following form

$$P(\theta) = \prod_{i=1}^{n} P(\theta_{X_i|\mathbf{Pa}_i})$$

∎

Definition 2.3.5: Let X be a variable with parents $\mathbf{U}$, we say the prior $P(\theta_{X|\mathbf{U}})$ has *local parameter independence* if $P(\theta_{X|\mathbf{U}}) = \prod_{\mathbf{u}} P(\theta_{X|\mathbf{u}})$ ∎

We say that the prior $P(\theta)$ satisfies *parameter independence* if it satisfies both global and local parameter independence.

In addition, Dirichlet priors are *conjugate priors*, that means that the posterior has the same functional form as the prior. This property provides an intuitive interpretation for the hyperparameters. One can view $\alpha_{x|\mathbf{u}}$ as imaginary counts, meaning prior to observing $\mathcal{D}$ we "observed" $X = x$ and $\mathbf{Pa}_x = \mathbf{u}$, $\alpha_{x|\mathbf{u}}$ times.

Proposition 2.3.6: *If $P(\theta)$ is $Dirichlet(\alpha_{x^1|\mathbf{u}}, \ldots, \alpha_{x^K|\mathbf{u}})$, then the posterior $P(\theta \mid \mathcal{D})$ is $Dirichlet(\alpha_{x^1|\mathbf{u}} + M[x^1, \mathbf{u}], \ldots, \alpha_{x^K|\mathbf{u}} + M[x^K, \mathbf{u}])$ where $M[x, \mathbf{u}]$ is the sufficient statistics derived from $\mathcal{D}$.*

Even when our prior is flat, the full affect of the Bayesian approach comes to play when predicting the probability of future samples. In the Bayesian approach the probability of a future observation is not calculated based on only one set of parameters, but using the expectation over the entire distribution of parameters. Thus the probability of a new sample $X[M+1]$ is:

$$P(X[M+1] \mid \mathcal{D}) = \int P(X[M+1] \mid \mathcal{D}, \theta) P(\theta \mid \mathcal{D}) P(\theta) d\theta \tag{2.8}$$

When we use table CPDs and Dirichlet priors this integral has a closed form solution:

$$P(X_i[M+1] = x^i \mid \mathbf{Pa}_{X_i}[M+1] = \mathbf{u}, \mathcal{D}) = \frac{\alpha_{x^i|\mathbf{u}} + M[x^i, \mathbf{u}]}{\sum_j \alpha_{x^j|\mathbf{u}} + M[x^j, \mathbf{u}]} \tag{2.9}$$

2.3.2 Structure Learning

Previously, we showed how one can learn the Bayesian network parameters given a known structure $\mathcal{G}$, but in reality, we do not know $\mathcal{G}$. Our goal is to understand the structural relationships between the variables in our domain, for instance we would like to be able to distinguish between a direct and indirect dependency between genes. Therefore, it is exactly this graph structure which we wish to reconstruct from the observed data. We can then use this reconstructed graph structure to answer queries regarding the interactions between the genes in our domain and other structural properties. We note that based on observational data alone, it is not possible to distinguish between equivalent structures (see Section 2.2.2). Thus, at best our reconstruction procedure can reconstruct an equivalence class of networks.

We take a *score-based approach* to this problem. We define a model space of candidate models which we are willing to consider and a scoring function that measures how well each model fits the observed data. Then we use an optimization algorithm that searches for the highest scoring model.

Bayesian Score

Our scoring function is based on the same Bayesian principles described in Section 2.3.1. The basic principle is: whenever we have uncertainty over something, we place a probability distribution over it. We therefore define a structure prior $P(\mathcal{G})$ over the different graph structures and a parameter prior $P(\theta \mid \mathcal{G})$ over the parameters once the graph is given. The particular choice of the priors $P(\mathcal{G})$

and $P(\theta \mid \mathcal{G})$ determine the exact Bayesian score. Our score evaluates the posterior probability of the graph given the data:

$$\text{score}_{\mathcal{B}}(\mathcal{G} \ : \ \mathcal{D}) \quad = \quad \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

where $P(Data \mid \mathcal{G})$ takes into consideration our uncertainty over the parameters and averages the probability of the data over all possible parameter assignments to $\mathcal{G}$.

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathcal{G}, \theta) P(\theta \mid \mathcal{G}) d\theta$$

The Bayesian is well suited for situations in which the number of samples is small. The Bayesian score is biased to more simple structures, but as it gets more data, it will support far more complex structures (when the generating distribution is indeed complex). This bias is due to the integration over all possible parameters. Structures with many parameters are penalized, unless the probability of the true parameters is very peaked (which happens when the sample size is large). Thus the Bayesian score inherently takes care of the problem of over-fitting a small sample to a complex model. In the rest of this section we show how to choose good priors and demonstrate how these priors lead to desirable properties in our score.

An important characteristic of the Bayesian score is that when we restrict ourselves to a certain class of *factorized* priors [18, 46] then the Bayesian score decomposes.

Definition 2.3.7: A parameter prior satisfies *parameter modularity* if for any two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, if $\mathbf{Pa}_i^{\mathcal{G}_1} = \mathbf{Pa}_i^{\mathcal{G}_2}$ then:

$$P(\theta_{X_i|\mathbf{Pa}_i^{\mathcal{G}_1}} \mid \mathcal{G}_1) = P(\theta_{X_i|\mathbf{Pa}_i^{\mathcal{G}_2}} \mid \mathcal{G}_2)$$

that means the parameter prior depends only on the local structure of the graph.∎

Proposition 2.3.8: *If the prior $P(\theta \mid \mathcal{G})$ satisfies global parameter independence and parameter modularity then*

$$P(\mathcal{D} \mid \mathcal{G}) \quad = \quad \prod_i \int_{\theta_{X_i|\mathbf{Pa}_i}} \prod_m P(x_i[m] \mid \mathbf{pa}_i[m], \theta_{X_i|\mathbf{Pa}_i}) P(\theta_{X_i|\mathbf{Pa}_i}) d\theta_{X_i|\mathbf{Pa}_i}$$

Therefore we can decompose the score into the local contributions of each variable (denoted FamScore), where the contribution of every variable X_i to the total network score depends only on the sufficient statistics of X_i and its parents $\mathbf{Pa}_i$.

$$\text{score}_{\mathcal{B}}(\mathcal{G} \ : \ \mathcal{D}) = \sum_i \text{FamScore}_{\mathcal{B}}(X_i, \mathbf{Pa}_i \ : \ \mathcal{D}) \tag{2.10}$$

As we will see in Section 2.3.2, this decomposition plays a crucial rule towards our ability to devise efficient search algorithms for high scoring network structures.

In this thesis we use Dirichlet priors for table CPDs. One of the big advantages of using Dirichlet priors is that the family score has a simple closed form formula. We present the formula itself and a sketch of how this formula is derived.

Theorem 2.3.9: [46] Let $\mathcal{G}$ be a network structure and $P(\theta \mid \mathcal{G})$ be a parameter prior satisfying parameter independence. Further assume table CPDs and Dirichlet priors with hyperparameters $\{\alpha_{X_i^j|u}\}$ then:

$$\mathrm{FamScore}_{\mathcal{B}}(X_i, \mathbf{Pa}_{X_i} : \mathcal{D}) = \log \prod_{\mathbf{u} \in Val(\mathbf{Pa}_{X_i})} \frac{\Gamma(\alpha_{x_i|u})}{\Gamma(\alpha_{x_i|u} + M[\mathbf{u}])} \prod_{x_i^j \in Val(X_i)} \frac{\Gamma(\alpha_{x_i^j|u} + M[x_i^j, \mathbf{u}])}{\Gamma(\alpha_{x_i^j|u})} \tag{2.11}$$

where Γ is the Gamma function and $\alpha_{x_i|u} = \sum_{j \in Val(X_i)} \alpha_{x_i^j|u}$

Proof: (sketch) Using the chain law we can view the Bayesian score as calculating the probability of each data sample given the previous ones. Using parameter independence assumptions we decompose the marginal likelihood into a separate term for each combination of a variable and possible assignments to its parents. Each such term can be solved in an independent manner.

$$\begin{aligned} P(\mathcal{D} \mid \mathcal{G}) &= \prod_{m=1}^{M} P(X[m] \mid X[1], \ldots, X[m-1], \mathcal{G}) \\ &= \prod_{i=1}^{n} \prod_{\mathbf{u} \in Val(\mathbf{Pa}_{X_i})} \prod_{m:\mathbf{Pa}_{X_i}[m]=\mathbf{u}} P(X_i[m] \mid X_i[1], \mathbf{Pa}_{X_i}[1], \ldots, X_i[m-1], \mathbf{Pa}_{X_i}[m-1], \mathcal{G}) \end{aligned}$$

By applying Eq. (2.9) and after some algebraic manipulation we get the desired formula.

∎

A desired property is that the score reaches its optimum on the true generating structure, i.e., given a sufficiently large number of samples, graph structures that exactly capture all dependencies in the distribution, will receive, a higher score than all other graphs (see for example [39]). This means, that given a sufficiently large number of instances, learning procedures can pinpoint the exact network structure modulo the correct equivalence class.

Definition 2.3.10: Assume our model is generated by some true model $\mathcal{G}^*$. We say that our score is *consistent* if as $M \to \infty$, the following properties hold with probability asymptotic to 1 (over possible choices of dataset $\mathcal{D}$)

- The structure $\mathcal{G}^*$ will maximize the score
- All structures that are not equivalent to $\mathcal{G}^*$ will have a strictly lower score

∎

Theorem 2.3.11: The Bayesian score is consistent

Recall that given $\mathcal{D}$ it is impossible to distinguish between two different networks in the same equivalence class. Thus a another desirable property in our score is that equivalent structures receive the same score, i.e., if $\mathcal{G}_1$ and $\mathcal{G}_2$ are equivalent graphs they are guaranteed to have the same posterior score. Such a property is called *structure equivalence*. In order to achieve structure equivalence, we devise a set of hyperparameters so that our prior will not bias the score between equivalent structures. This is achieved using a *BDe prior* [45]. We define a probability distribution P' over $\mathcal{X}$ and an equivalent sample size M' for our set of imaginary samples. The hyperparameters are then defined to be:

$$\alpha_{x_i|u_i} = M' \cdot P'(x_i, u_i)$$

Theorem 2.3.12: [46] When the data is complete, the prior satisfies parameter independence and parameter modularity, and Dirichlet BDe priors are used, then the score is structure equivalent.

Proof: As a consequence of Theorem 2.2.8 it is enough to prove that the score is equal for any two equivalent graphs that differ only in the orientation of a single covered edge. Let $\mathcal{G}^1$ and $\mathcal{G}^2$ be two such graphs and $X - Y$ the edge that they differ on. Because the two graphs differ only in the reversal of a single edge, the only terms in Eq. (2.10) that differ between the two graphs are $\mathrm{FamScore}_{\mathcal{B}}(X, \mathbf{Pa}_X \ : \ \mathcal{D})$ and $\mathrm{FamScore}_{\mathcal{B}}(Y, \mathbf{Pa}_Y \ : \ \mathcal{D})$. We denote $U := \mathbf{Pa}_X^{\mathcal{G}^1}$. Since $X \rightarrow Y$ is covered in $\mathcal{G}^1$ this means that $\mathbf{Pa}_Y^{\mathcal{G}^1} = U \cup X$, $\mathbf{Pa}_X^{\mathcal{G}^2} = U \cup Y$, and $\mathbf{Pa}_Y^{\mathcal{G}^2} = U$. Recall, the hyperparameters for the BDe prior are defined to be $\alpha_{x|u} = M' \cdot P'(x, u)$ where P' is a prior distribution over $\mathcal{X}$ and M' is the equivalent sample size.

We denote

$$l(\mathbf{X}) \ = \ \prod_{\mathbf{x} \in Val(\mathbf{X})} \Gamma(M' \cdot P'(\mathbf{x}) + M[\mathbf{x}])$$

using this notation

$$\mathrm{FamScore}_{\mathrm{Int}}(X, \mathbf{Pa}_X^{\mathcal{G}^i} \ : \ \mathcal{D}) = \frac{l(X \cup \mathbf{Pa}_X)}{l(\mathbf{Pa}_X)}$$

Using these notations, in order to prove the equivalence between the two scores we need to show that the following two expressions are equal: The contribution of X and Y to the score for $\mathcal{G}^1$ is:

$$\frac{l(X \cup U)}{l(U)} \frac{l(X \cup Y \cup U)}{l(X \cup U)}$$

whereas their contribution to the score for $\mathcal{G}^2$ is:

$$\frac{l(X \cup Y \cup U)}{l(Y \cup U)} \frac{l(Y \cup U)}{1(U)}$$

The score for $\mathcal{G}_1$ and $\mathcal{G}_2$ is equal. ∎

Search Algorithm

Once the score is specified and the data is given, learning amounts to finding the structure G that maximizes the score. This problem is known to be NP-hard [12]. Thus, we resort to a heuristic search. We define a *search space* where each *state* in this space is a network structure. We define a set of *operators* that take us from one structure to another. This defines a graph structure on the states: neighboring states are those which are one operator away. We start with some initial structure (usually the empty graph) and using the operators traverse this space searching for high scoring structures.

A natural choice of neighboring structures are a set of structures that are identical to the base structure except for *local* modifications. We use the following operators that change one edge at each move:

- Add an edge
- Remove an edge
- Reverse an edge.

Note we only consider operations that result in legal networks. That is a-cyclic networks that satisfy any other constraints we specify (e.g. maximal indegree constraints).

A *local* search procedure can efficiently evaluate the gains made by adding, removing or reversing a single edge. The decomposition of the score is crucial for the efficiency of this procedure. Due to the score decomposition, we only need to re-evaluate those components of the score that involve the variables affected by our local step. For instance, if we add an edge to the variable X_i, we only have to recalculate X_i's family score, the family scores for all other variables remain unchanged.

We use a greedy hill-climbing algorithm for our search procedure: At each step we evaluate all possible local moves and perform the change that results in the maximal gain, until we reach a local maximum. (The pseudo-code for such an algorithm appears in Figure 2.5) Although this procedure does not necessarily find a global maximum, it does perform well in practice. A number of heuristics can be included to overcome some of the local maxima: these include random restarts and Tabu search. Examples of other search methods that advance using single edge changes include beam-search, stochastic hill-climbing, and simulated annealing.

Any implementation of these search methods involves caching of computed counts to avoid unnecessary passes over the data. This cache also allows us to *marginalize* counts. Thus, if $M[X, Y]$ is in the cache, we can compute $M[X]$ by summing over values of Y. This is usually much faster than making a new pass over the data. One of the dominating factors in the computational cost of learning from complete data is the number of passes actually made over the training data.

Input:
$\mathcal{D}$ // A data set
$\mathcal{G}_0$ // Initial network structure
Output:
$\mathcal{G}$ // Final network structure
Greedy-Structure-Search
$\mathcal{G}_{best}$ = $\mathcal{G}_0$
repeat // apply best possible operator to $\mathcal{G}$ in each iteration
$\mathcal{G} = \mathcal{G}_{best}$
foreach operator o // (each edge addition, deletion or reversal on $\mathcal{G}$)
$\mathcal{G}^o = o(\mathcal{G})$ // apply o to $\mathcal{G}$
If $\mathcal{G}^o$ is cyclic **continue**
If $\text{score}_{BDe}(\mathcal{G}^o \ : \ \mathcal{D}) > \text{score}_{BDe}(\mathcal{G}_{best} \ : \ \mathcal{D})$
$\mathcal{G}_{best} = \mathcal{G}^o$
until $\mathcal{G} == \mathcal{G}_{best}$ // no change in structure improves score

Figure 2.5: Outline of Greedy search algorithm

Chapter 3

Bayesian Network Models for Biological Interactions

3.1 Overview

Molecular pathways form a complex web of interacting proteins, genes, and small molecules. The type of interactions is varied and includes signal transduction and processing, regulation of gene expression and metabolism. Our goal is to directly infer the fine structure of such interactions from raw experimental data such as gene expression profiles. In this chapter we employ Bayesian network learning techniques to reconstruct models of biomolecular systems from the observed measurements with a goal to gain insight into the structure of molecular interactions. We hope to answer questions such as: is the effect of an effector gene on a target gene direct, or is it mediated by other genes? And which genes mediate between them?

We adopt a *systems* perspective of the cell and its components. Our goal is to estimate the joint probability distribution over gene expression and understand its structural features from data. Our reconstruction of pathway structure is based on the following idea: molecular interactions between the genes generate corresponding statistical dependencies between the random variables that represent them. We model statistical dependencies using *Bayesian networks*. Our learning algorithm detects consistent dependencies and reconstructs a Bayesian network that best explains the observed data. Our approach is global in that: we fit a model to data by studying the joint probability distribution over the entire gene set.

Note that a true biological network is not a Bayesian network: for instance Bayesian networks are acyclic while cyclic and feed-back loops are central in biological networks and their regulatory mechanisms. Still, using Bayesian networks as a first order approximation, many interesting molecular interactions emerge.

The key advantage of using Bayesian networks is they model the joint distribution over a number of variables at once. Most standard methods for analyzing gene expression focus on pairwise relations between genes (e.g. correlation). However, biological interaction is seldom this simple

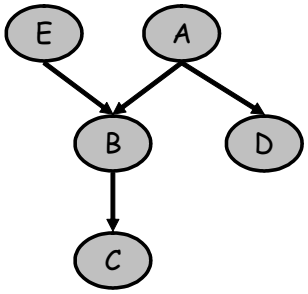

Figure 3.1: An Bayesian network representing a pathway with 5 genes

and often includes chains of mediators between two genes. By exploring multi-variate interactions, we can infer more structure in the relationship between genes. We will focus on *conditional independence*: for example, if X and Y are co-regulated by Z, then while Y correlates with X, given the value of Z, Y might become independent of X. In this case, we say that Z *separates* between X and Y.

Several important issues arise when learning a Bayesian network model in the gene expression domain. Most of the difficulties in learning from expression data revolves around the following central point: contrary to most situations where learning a models is attempted (in particular Bayesian networks), expression data involves transcript levels of thousands of genes, while currently available data sets contain at most a few hundred samples. This paucity of data raises issues with both computational complexity of the learning algorithm and the statistical significance of the resulting networks. It is generally believed that genetic regulation networks are sparse in the sense that, for a given gene, it is assumed that no more than a few dozen genes directly effect its transcription. Since Bayesian networks are especially suited for learning in such sparse domains , we believe that it is appropriate to them in our context.

3.2 Illustrative Example

In this section we demonstrate how we can use a Bayesian network to model a molecular pathway and provide some intuition as to how such a model might be inferred from the data. Our example pathway consists of five genes, A,B,C,D,E; and the Bayesian network model contains five random variables representing the expression of each of these genes (see Figure 3.1) that can be in one of three states: down regulated, no change, and up regulated.

Gene A is a transcription factor that activates B, and in most cases when A is up-regulated so is B. If A activates B we expect the data to express a dependency between the values of A and B and we indicate this dependency by drawing a directed edge from A to B. The expression of A and B are statistically dependent, thus, knowing the value of A provides *information* that can help predict the value of B.

In addition, gene B is a transcription factor that activates gene C, generating an edge from B

to C in our network model. Based on pairwise data correlations, we expect the expression of C to be correlated not only with its direct regulator (B), but also by its indirect regulator (A). In our example, if we know the value of B, A does not provide additional information that can improve our predictions for C, and we say "the effect of A on C is *mediated* through B". In Bayesian network terms "A and C are *conditionally independent* given B". We might infer such mediation using knock-out data, i.e, while under normal conditions A and C are correlated, when gene B is deleted, this correlation disappears.

Gene A also activates gene D. This provides us with another example of conditional independence. Taking a pairwise view, D is a good predictor of B's expression - we reason that D's up-regulation might be caused by A's up-regulation and therefore B is more likely to be up-regulated as well. Notice, that in this chain of reasoning, A *explained* the correlation between the expression of B and D. It is exactly this type of *explaining away* that we hope to achieve by using a multivariate view of the data. How might we infer this directly from the data? For instance, D might be up-regulated from causes other than A's up-regulation. Thus, we hope to find that A is a much more reliable predictor for B than D.

Finally, gene E inhibits gene B. Thus, B has two different genes that control its regulation. These are B's *parents* in the Bayesian network. This leads us to the second component in the Bayesian network: each gene has a local conditional probability model that describes the probability for each of its states conditioned on its parents values. The probabilistic nature of the function between B and its parents has several advantages:

1. The biological processes involved are inherently stochastic.
2. Our measurements are noisy.
3. There might be other unknown factors not included in our measurements that effect the expression of B, a probabilistic function can treat such an uncertainty.

Of course, real life is not that simple - a molecular pathway is far more complex, consisting of thousands of interacting genes. Current gene expression datasets are very noisy, and automatically inferring such a model directly from data is very challenging task. The biggest complication stems from the fact that the gene expression is effected by countless other factors other than the expression of other genes. These factors are not included in our observed measurements.

3.3 Extracting Features

Given a dataset of expression profiles, we use a learning algorithm to search for the Bayesian network $\mathcal{G}$ that best explains the data (see Section 2.3). The simple minded approach would be to accept $\mathcal{G}$ as a correct model of our domain. Then we could use $\mathcal{G}$ to infer structural relations between genes (e.g. gene A directly effects gene B). Such anaysis would rely on the assumption that the network $\mathcal{G}$ correctly represents the interactions in the underlying domain, but how reasonable is

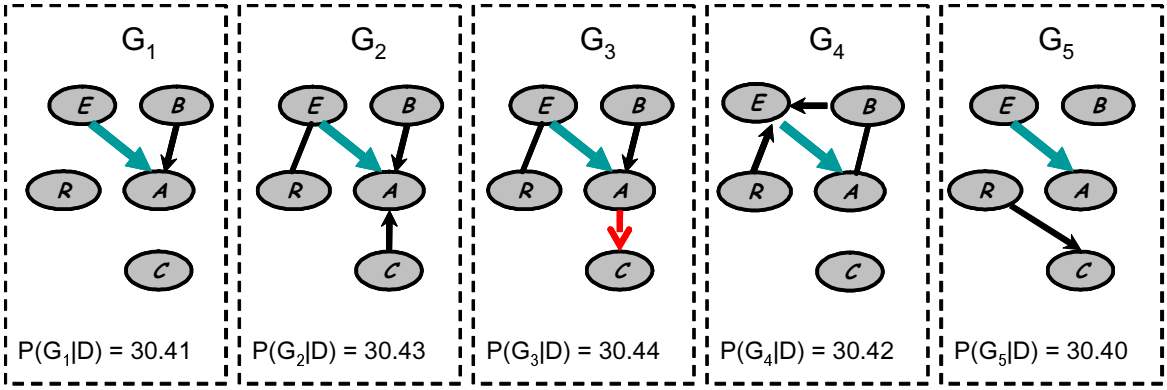

Figure 3.2: An ensemble of high scoring networks: Presented are five different Bayesian networks. The score of the network with respect to the dataset D is listed below each network. While $\mathcal{G}_3$ is the highest scoring network, the other networks are almost as likely. The edge $A \rightarrow C$ exists in $\mathcal{G}_3$, but it is absent from all other high scoring networks. On the other hand, all five networks agree on the edge $E \rightarrow A$.

this assumption? A sufficiently large number of samples would (almost) guarantee that it is certain that the network we learn is a good model of the data [39]. However, given only a small number of training instances, there may be many models that explain the data almost equally well among themselves.

Figure 3.2 shows five high scoring networks in relation to dataset D. While the structure of the networks significantly vary, their scores are almost the same. In our example, $\mathcal{G}_3$ is the highest scoring network, but by removing the first sample from D, $\mathcal{G}_4$ becomes the highest scoring network. A simple minded approach, we would infer a direct interaction between the genes A and C (since the edge $A \rightarrow C$ exists in $\mathcal{G}_3$). But, the edge is absent from the other high scoring networks. Therefore, it is more likely that the dependence between A and C is a spurious artifact in D. Thus, we cannot depend on a one single network to provide an accurate description of the relations in our biological domain.

Instead of querying a single structure, we search for common *features* which most of the high scoring network structures agree on. For example, the ensemble of high scoring networks in Figure 3.2 all agree on the edge $E \rightarrow A$. Therefore, it is highly likely to represent a real biological signal. We find such features by examining the *posterior* probability of the feature given the data. A network *feature* is a property such as "$X \rightarrow Y$ is in the network" or "*d-sep*$_{\mathcal{G}}(X; Y \mid \mathbf{Z})$ is in the network". We associate the feature f with an indicator function, $f(\mathcal{G})$, that has the value 1 when $\mathcal{G}$ satisfies the feature and value 0 otherwise. The posterior probability of f is defined as:

$$P(f(\mathcal{G}) \mid D) = \sum_{\mathcal{G}} f(\mathcal{G}) P(\mathcal{G} \mid D). \tag{3.1}$$

This probability reflects our *confidence* in f, given D.

The straingt forward way of calculating Eq. (3.1) is by enumerating all high scoring networks. Unfortunately, the number of such networks can be exponential in the number of variables, so exact computation of the posterior probability is not feasible. Instead, we can estimate this posterior by sampling representative networks and then estimating the fraction that contain the feature of interest. Ideally, we would like to sample networks from $P(\mathcal{G} \mid D)$ and use the sampled networks to estimate this quantity. The general solution to this problem is to build a *Markov Chain Monte Carlo* (MCMC) sampling procedure, such as that suggested by Friedman and Koller [35]. Unfortunately, this sampling procedure is also computationally costly.

Instead, we choose to use an effective, and relatively simple *bootstrap* method [29] as an approximation of the posterior probability. Our networks are generated using non-parametric bootstrap as suggested by Friedman and Goldzmidt [34]. The main idea behind the bootstrap is simple: We generate "perturbed" versions of the original data set and learn a Bayesian network structure from each of them. In this way we collect many networks, all of which are fairly reasonable models of the data. These networks reflect the effect of small perturbations to the data on the learning process. We use the following procedure:

- For $i = 1 \ldots m$.
  - Construct a dataset D_i by sampling, with replacement, M instances from D.
  - Apply the learning procedure on D_i to induce a network structure $\mathcal{G}_i$.
- For each feature f of interest calculate

$$\text{conf}(f) = \frac{1}{m} \sum_{i=1}^{m} f(\mathcal{G}_i)$$

We refer the reader to Section 3.7.1 for an evaluation of this bootstrap approach on simulated data. These simulation experiments show that features induced with high confidence are rarely false positives, even in cases where the data sets are small compared to the system being learned. We note that the rate of false negatives is high. Thus, the fact that we do not detect high confidence for a feature, does not mean it does not exist, but rather that the data does not strongly support it.

3.4 In Silico Experiment

In this section we present our overall framework for analyzing gene expression data with Bayesian networks. We illustrate our method using two datasets derived from the Compendium [51] expression profiles. Figure 3.3 provides an overview of our analysis method.

The first step is to preprocess the expression data. This includes selecting the relevant genes and distcretizing their expression values. We use two complementary approaches to gene selection. One method chooses genes that are highly variant in the data, based on the reasoning that such genes display potentially interesting behavior. The second approach is to focus on a specific biological

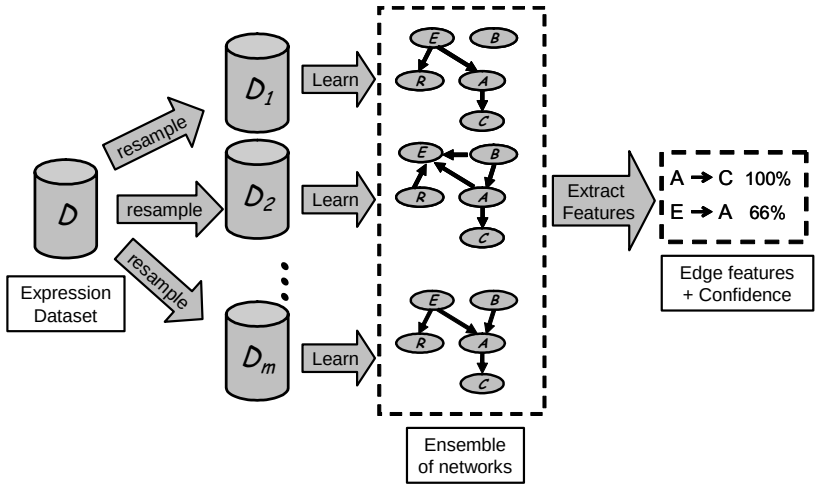

Figure 3.3: Outline of Bayesian network analysis of gene expression data

process of interest (e.g. mating) and choose genes that are associated with that process. After gene selection, each gene is discretized into one of three values: down regulated, no change, or up regulated. Our methods for discretization are dataset dependent.

We apply the non-parametric bootstrap described in Section 3.3 to the dataset of discretized gene expression measurements derived above to learn an ensemble of networks which represent potential models of the interactions between genes. We developed the *Sparse Candidate algorithm* (SPC) in order to deal with the computational complexity of learning Bayesian networks over many variables. Each time Bayesian network learning is invoked, we use the sparse candidate algorithm and convert the resulting network $\mathcal{G}$ into its associated PDAG $\mathcal{P}$. Details of SPC algorithm are provided in Section 4.1.

We use the ensemble of Bayesian networks which we learned in order to extract statistically confident features involving relationships between pairs and triplets of genes. Finally, we statistically identify significant subnetworks that are dense in high-confidence features. Identifying subnetworks will be described in Section 3.6.

As a case study, we apply our framework to the Rosetta Inpharmatics Compendium [51] of expression profiles from *Saccharomyces cerevisiae*. This dataset is compiled from 300 full-genome expression profiles obtained from 276 deletion mutants, 11 tetracyclin regulatable alleles of essential genes, and 13 chemically treated *S. cerevisiae* cultures, each compared to a baseline wild type or mock-treated culture. This dataset is well suited for our methods because the different deletions and treatments perturb the data, thus creating a wide variety of different cell states. This diverse set of samples captures which genes remain dependent on each other across many different conditions and how these dependencies behave.

The variance-oriented dataset is composed of 565 genes and includes the mutated genes, as well as genes which showed a significant change in at least 4 profiles (this dataset was used in Pe'er

et al. [76]). The discretized dataset is available at URL. Since the samples in the dataset include mutations and the gene set includes the mutated genes, we used a correction to the scoring function that takes these mutations into account. When a gene is mutated, it is disassociated from its natural regulatory mechanism. Instead, an external mechanism controls the expression of the gene. This event is specified in our model by disconnecting the edges between the mutated gene and its parents in the Bayesian network structure $\mathcal{G}$. (see Section 4.2.

The subnetworks constructed from the variance-oriented dataset include parts of the mating and amino acid metabolism pathways. To evaluate how well our approach is capable of reconstructing these processes, we created a new dataset from the compendium profiles. The process-oriented dataset is composed of 947 genes, including mating and amino acid metabolism related genes, as well as genes which showed significant change in at least 4 profiles. The genes in this dataset had less variance across samples. The discretization procedure used for the variance-oriented dataset would classify many of these genes as unchanged across all samples, erasing all information on their behavior. Thus, we used a different discretization procedure to overcome this problem, the discretized dataset is available at URL. Since modeling the mutations require more intensive computation, we used the standard BDe score for this dataset, ignoring the mutations. The benefit of modeling mutations comes into play when asking causal queries and in this analysis of this dataset we mostly evaluated undirected pairwise relations.

We applied a 100-fold bootstrap resulting in an ensemble of 100 learned networks for each dataset. The networks learned from the variance-oriented dataset contained an average of 1243 edges each (an average of 2.2 parents for each gene). The networks learned from the process-oriented dataset contained an average of 2652 edges each (an average of 2.8 parents for each gene). While the learned networks are relatively rich in interactions, as we will see in Section 3.5, relatively few of them remained consistent across many networks. In the following sections we provide an analysis of those interactions that remained consistent across networks. Interestingly, they correspond well with known biology.

3.5 Biological Features

In this section we focus on the following question: How can our analysis elucidate the nature of interaction between genes? We consider several types of "features" that can be identified from $\mathcal{G}$ and $\mathcal{P}$ and discuss possible biological interpretations of such features.

3.5.1 Gene Mates

The first type of feature is *gene mates*, also called *Markov relations* in [34, 36, 76]. This feature focuses around the question: Do X and Y directly interact in our network. We answer this by querying $\mathcal{G}$ to test whether X and Y are *Markov neighbors* [73]. Markov neighbors are variables that are not separated by any other measured variables in the domain, i.e. no subset of variables in

the domain render them independent. They include parent-child relations (e.g. one gene regulating another), or spouse relations (e.g. two genes that co-regulate a third). Note that two gene mates are directly linked in the sense that no variable *in the model* mediates the dependence between them. It remains possible that an unobserved variable is an intermediate in their interaction. Since our domain involves many factors that are not modeled into our network (e.g. protein activation), many of the resulting gene mates represent two genes that are regulated by a third *latent* [32] factor. Note, that while gene mates is a pairwise relation, it is based on multivariate analysis. Before the learning procedure assigns a gene mate relation to a pair X and Y, it searches for a subset $\mathbf{Z}$ so that given $\mathbf{Z}$, the genes X and Y become conditionally independent. Thus, to derive a gene mate relation, a series of statistical tests involving a larger set of variables is performed. The relations reported here often included statistical queries involving 3 to 8 different variables.

We can query whether the edge $X \rightarrow Y$ appears in $\mathcal{P}$. Recall, that this implies that X and Y are Markov neighbors (parent-child type) and that the edge between them is directed in all networks in the equivalence class of $\mathcal{G}$. The existence of such a directed edge might suggest that X is a direct *cause* of Y. One must treat causal conclusions derived from a Bayesian Network with caution as they involve a number of assumptions that do not always hold in our domain (See [74, 19] for the connection between Bayesian networks and causality).

Biological analysis of individual high confidence gene mate relations indicates that many are supported by previous biological findings. To understand the resulting gene mates, we ask the question "What biological relation(s) can lead to a direct statistical dependency between two genes in the expression data?" There are a variety of biological interactions that can create such a direct statistical dependency, thus gene mates do not have unique biological semantics. Gene mates can represent different regulatory relations between genes (both direct and indirect), co-regulation, physical protein-protein interactions (such as signaling cascades and complexes) and metabolic links. Often, these links are mediated through one or more hidden factors, unobserved in expression data.

We will illustrate some of the different types of gene mates, providing an example for each from our analysis of the process-oriented dataset. For each gene mate, we calculate the confidence (i.e. the percent of the networks in which the relation exists) and the Pearson correlation between the raw expression profiles.

Regulation: Perhaps the simplest type of interaction is a transcription factor X that regulates a target Y. We found 6 such relations, for example: Met28 $\rightarrow$ Sul2 (confidence=0.83, Pearson correlation=0.72). Met28 is a transcriptional activator of sulfur amino acid metabolism that regulates Sul2, a sulfate transporter.

Signalling: Surprisingly, four of the gene mates involved signalling proteins. For example, Slt2 $\rightarrow$ Crh1 (confidence=0.98, Pearson correlation=0.56). Slt2 is a MAP kinase that activates cell wall related processes and it indirectly regulates Crh1, a cell wall protein. Slt2 is known to activate Rlm1, which in turn is known to bind to Crh1, thus the correct biological regulatory chain would be Slt2 $\hookrightarrow$ Rlm1 $\hookrightarrow$ Crh1. At first, one might think that our method should have captured that Slt2's effect on Crh1 is "mediated" via Rlm1, but this is a good example of the limits of expression

data. Slt2 regulates Rlm1 post-translationaly, an event that is "hidden" in our domain. When post-translationally activated, Rlm1 does not show any observable change in its expression level.

Co-Regulation: Many gene mates are simply pairs of co-regulated genes. For instance, Scw11 – Dse2 (confidence=0.98, Pearson correlation=0.86). Both Scw11 and Dse2 are genes involved in cytokinesis: The process of finishing cell separation, resulting in two physically separated cells. Both genes are known to be regulated by Ace2, a transcription factor involved in G1-specific transcription in mitotic cell cycle. Thus the correct biological network would be Scw11 $\hookleftarrow$ Ace2 $\hookrightarrow$ Dse2. Again, one could question why Ace2's role in the network was missed. While Ace2 is transcribed in the G2 phase of the cell-cylce, it is only found in the nucleus in late M and early G1 phases, and therefore only then active. The post-translational regulation of Ace2's location is not observed in the gene expression data. Since the change in Ace2's regulatory status is undetectable in the gene expression data, there is no variable that can explain away the dependency between Scw11 and Dse2. When Ace2 is active the expression of both Scw11 and Dse2 react in the same fashion and are thus dependant on each other.

Metabolic Step: There are 29 gene mates whom are close to each other in a metabolic pathway, often only one metabolic step away, i.e., they share a common metabolite. Many of these pairs are co-regulated by general metabolic regulators (e.g. Gcn4), thus we could simply categorize these pairs as co-regulation. Still, the number of pairs which are proximately very close along a metabolic pathway is a striking phenomenon. For example, Met3 – Met14 (confidence=0.96, Pearson correlation=0.87). Met3 and Met14 are two consecutive genes along the metabolic pathway that reduces sulfate to sulfide. We conjecture that genes which are close on the same metabolic pathway are more closely co-regulated than a pair of metabolic genes which are less related. We elaborate on metabolic gene mates in Section 3.7.

Complex: There are 4 gene mates that belong to the same protein complex. For example, Sdh2 – Sdh3 (confidence=0.75, Pearson correlation0.29). Sdh2 and Shd3 are both part of a four protein succinate dehydrogenase complex involved in oxidative phosphorylation. As with the metabolic step, we can speculate that proteins that form a complex have a tighter co-regulation mechanism and are thus directly dependant in the Bayesian network. But, this tighter co-regulation does not necessarily manifest itself as tighter correlation. Notice that while the confidence of this link is high (0.75), the Pearson correlation of the expression of these two genes is very low (0.29). This demonstrates that Pearson correlation is not always a good measure of co-regulation. In the case of this pair, the two genes are highly correlated in their extreme behaviors (strongly up or down regulated), but have no apparent correlation across most arrays in which the two genes are in their basal behavior. Thus, we can speculate that these genes are tightly co-regulated only under specific conditions when the complex needs to be closely controlled.

Related Function: Some genes mates do not have any known co-regulator, but both have a related function or process. In this case, we assume it is likely that these genes share a common yet unknown cause that is not necessarily a regulatory gene. Such a common cause could be an external signal that might activate two distinct gene regulatory chains. An example of functionally

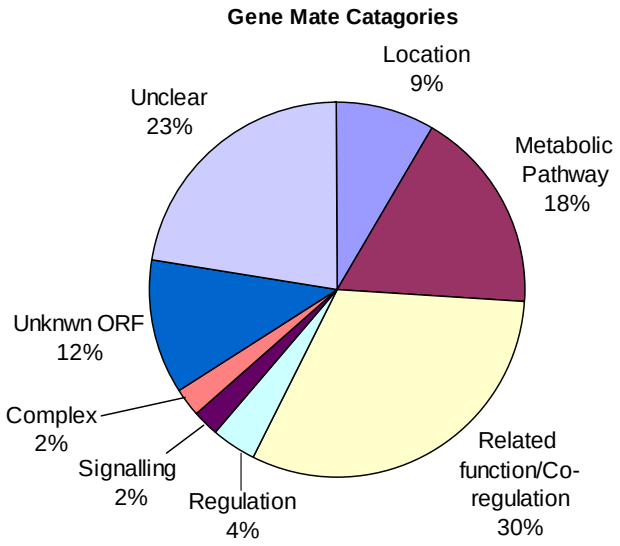

Figure 3.4: Categories of gene mates: Analysis of Dataset II resulted in 164 gene mates of confidence > 0.75. This pie-chart represents the distribution between the different categories

related genes is Sno2 – Sno3 (confidence=0.98, Pearson correlation=0.4). Both genes are induced in stationary phase and both participate in the processes of pyridoxine metabolism and thiamin biosynthesis. As in the previous example, the Pearson correlation between genes is low and therefore while these genes share a common function, they would not be clustered together using a typical clustering algorithm. The gene expression of this pair is only correlated under conditions in which they are highly induced (i.e. under conditions which we believe they might be co-regulated), but have little correlation otherwise.

Location: There is a surprisingly high number of gene mates (14) that are consecutive along the chromosome. While some of these might be artifacts due to cross-hybridization, many of the examples, such as YLR311C – YLR312C (confidence=0.89, Prearson correlation=0.52), have low Pearson correlation making cross-hybridization unlikely. Thus we speculate that in a similar manner to the regulons in prokaryotes, genes of close physical proximity along the chromosome might share common regulatory mechanisms [7]).

In Figure 3.4 we evaluated and categorized all gene mates with confidence greater than 0.75 in the process-oriented dataset. It is evident that 2/3 of the relationships have a clear biological explanation. Furthermore, we observe that most of the gene mates are due to some mechanism of co-regulation and only a small portion of the gene mates are regulatory, i.e., one gene regulates the other. Finally, the large portion of gene mates with close metabolic proximity or close chromosomal proximity is evident.

About 1/3 of these relations had Pearson correlation less than 0.5. This shows that many genes are co-regulated only under specific conditions. Thus, co-expression is not always a good measure

for co-regulation: illustrating the differences between our techniques and clustering methods. Our methods are able to discover inter-cluster interactions between weakly correlated genes, while at the same time, uncovering finer intra-cluster structure among correlated genes (e.g metabolic links, regulatory relations, and protein-protein interactions). This assists us in understanding the roles of genes within a richer context.

3.5.2 Separators

Gene expression data usually contains many clusters of highly correlated genes, yet in our analysis only few confident gene-mate relations are formed. Often, correlated genes that do not pair are no less illuminating than the genes who do pair together. When X and Y are indirectly dependent, we can ask what other factors *mediate* this dependence. In the simple case, a single variable Z, separates X and Y in $\mathcal{G}$. For example, the edges $X \rightarrow Z \rightarrow Y$ or the undirected edges X—Z—Y appear in $\mathcal{P}$. In the former, X effects Z, which in turn effects Y; while in the latter, Z might be a common cause of both X and Y. The most interesting cases are those in which the same variable Z consistently separates X and Y in many of the high scoring networks. When this occurs, we define Z to be a *separator* between X and Y.

To illustrate the concept of a separator and its underlying context within the molecular architecture of pathways, we present two examples from the variance-oriented dataset: The first separator, Kar4, is a mating transcriptional regulator of karyogamy (nuclear fusion) genes. Kar4 separates several pairs of cell fusion genes (e.g. Aga1 and Fus1). From the semantics of the Bayesian network we conclude that the gene expression of Aga1 and Fus1 is correlated because they have a common regulator, Kar4. This is not a perfect match with the biological literature, in which Kar4 is known to regulate nuclear fusion. Since only a small part of the biology is currently known, we allow ourselves to make the hypothesis that Kar4 might have an additional role in the regulation of cellular fusion.

Sst2 is a post-translational negative regulator of the G-protein in the mating signaling pathway. In our inferred networks, Sst2 separates the mating response genes Tec1 and Ste6. Moreover, a high confidence directed edge (in $\mathcal{P}$) was discovered from Sst2 to Ste6, providing further support for the causal direction of the relationship. In this case our method captured a signalling regulator which explains the dependency of the expression of the genes it regulates. Thus, our inference has reconstructed the regulatory role in the correct molecular and functional context for both transcriptional and post-translational regulators.

In more complex cases, X and Y may be more distant in the graph structure (e.g Z is a common grandparent of both X and Y) and there might be more than one variable that mediates their interaction (e.g X is parent of Z_1 and Z_2, who in turn are both parents of Y). In these cases we must employ a global approach, searching for a set of variables $\mathbf{Z}$, such that Y is independent of X given $\mathbf{Z}$. We test that X and Y are independent given $\mathbf{Z}$ using d-separation, i.e., no path between X and Y can "pass" information when the value of $\mathbf{Z}$ is known (See Section 2.2.1 for the precise definition). The complexity of computing d-separation for every pair of variables in the network, even

by a single variable, is $O(n^3)$. For a large domain, this calculation is time and memory consuming. We note however, that when two variables are far from each other in the network, the dependence between them diminishes. Thus, in practice, we only check for single variable d-separators between variables along paths of length at most six.

When applying this procedure to the high scoring structures of the variance-oriented dataset, we found 120 d-separator triplets (a d-separator and the two genes it separates). Strikingly, in 35 of the resulting 120 interactions, the d-separator was either a transcriptional or a post-translational (signaling) regulator[1]. Such genes were considerably less frequent in the role of the separated genes (29/240 genes). These results are consistent with a regulatory role of the mediating gene.

When applying this procedure to the process-oriented dataset, we found that the d-separator was a regulator in less than 1/10 of the triplets. We attribute this qualitative difference to the fact that the networks created from the variance-oriented dataset used an algorithm that incorporated mutations existing in the data to aid causal discovery. We speculate these mutations yielded extra causal information, enabling the algorithm to place the regulators in their correct role. Since the edges in the Bayesian network do not have clear biological semantics, interpretations for separator relations require further study.

3.5.3 Hubs

A global view of the separator triplets reveals that the same separator often appears in many triplets. These multi separators, called *Hubs*, form a star-like structure: a central separator gene connected to a set of genes that are rarely connected amongst themselves. When evaluating such a hub in its entirety, a clearer biological story often emerges. We illustrate this using a few examples from process-oriented dataset.

The first example is Slt2, which encodes the MAP kinase of the cell wall integrity pathway. At a confidence level greater than 0.4, Slt2 interacts with 12 gene mates. These have no interactions amongst themselves. The genes in the Slt2 hub include 6 cell wall and cell membrane proteins (Crh1, Yps1, Yps3, Yps6, Sed1 and Prm5), and 2 signalling proteins associated with cell wall pathways (Ptp2 and Rga1). Ptp2 is a protein phosphotase responsible for inactivating the MAPK osmolarity pathway which Slt2 activates. In addition, the hub contains two genes related to DNA metabolism (Srl3 and Ypr078c) and two uncharacterized genes (Ybr005w and Yhr209w). Cluster analysis [51] puts the cell wall genes and the uncharacterized genes in the same cluster, while Srl3 and Ypr078c are assigned to a different cluster. Thus, at first glance, Srl3 and Ypr078c seem unrelated to Slt2. DNA location analysis [62] reveals that Rlm1, one of Slt2 target transcription factors, binds to Srl3, supporting its inclusion in the hub.

Hub structures sharpen the difference between clustering and the Bayesian network approach. A typical clustering algorithm would cluster together 10 of the hub genes (see Figure 3.5.3) forming a

[1]The 8 transcriptional regulators include general repressors (Isw1 (2 relations), Top1(1), Sin3(5)), specific transcription factors (Mth1(1), Rgt1(1), Imp2(1)), and putative transcriptional regulators (Yfl052w(1), Ypr015c(1)). The 7 signaling molecules are Kss1(1), Mfa2(3), Ras1(6), Rho1(6), Ste11(1), Tfs1(3), Ykl161c(2).

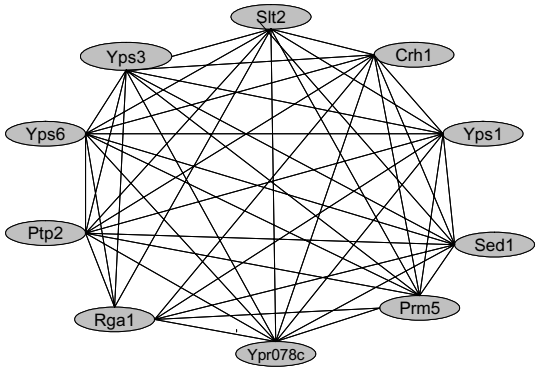

Figure 3.5: A clustering representation of the Slt2 hub: Each pair of genes in the cluster are connected. No gene has a distinct role.

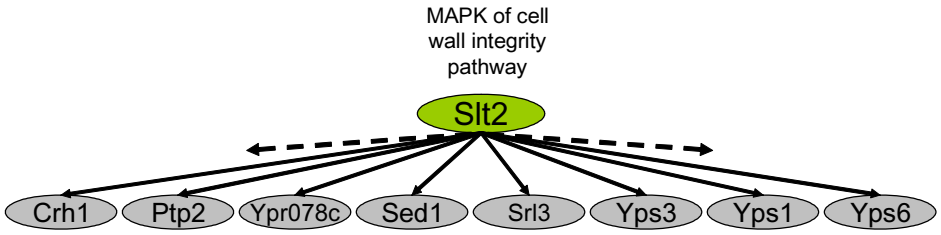

Figure 3.6: A Bayesian network representation of the Slt2 hub: A clearer interpretation emerges. Slt2 explains away the dependencies between its targets. Additional genes join the hub (Slr3).

dense network with no clear interpretation. In the Bayesian network, an explanation emerges: a set of cell wall genes are co-regulated by a common MAP kinase signaling protein (see Figure 3.5.3). Bayesian networks also provide a "cleaner picture". On one hand, while the Slt2 hub which contains almost solely cell wall genes, the Slt2 cluster contains many genes that are not related to cell-wall. On the other hand, related genes which are included into the hub (e.g. Srl3), are not identified by clustering. Within the context of a hub, we can confidently assign previously uncharacterized genes (Ybr005w and Yhr209w) with a putative effector function in cell wall integrity.

Another hub example is Ste2: the first step in the mating pathway in which the mating pheromone binds to its receptor. Ste2 is the pheromone receptor for the alpha factor. Ste2 is interacts with 6 gene mates (none of which interact with each other), Figure 3.5.3 depicts the known molecular pathway associated with these genes. These include 5 genes which are directly involved with mating factors (Ste4, Ste6, Mfa1, Mfa2 and Bar1) and Far1. All 6 effector genes are known to be involved with mating signalling. The hub gene activates this response after detecting the incoming signal (alpha mating factor). In a signalling cascade, Ste2 directly activates Ste4 via a direct protein-protein link. All hub genes are known to be transcriptionally regulated by this pathway via the transcription factor Ste12. Ste6, Mfa1 and Mfa2 are involved in creating and secreting the a-factor mating signal. Bar1 inactivates the alpha factor. Finally, Far1 is a cell cycle protein important for mating, through

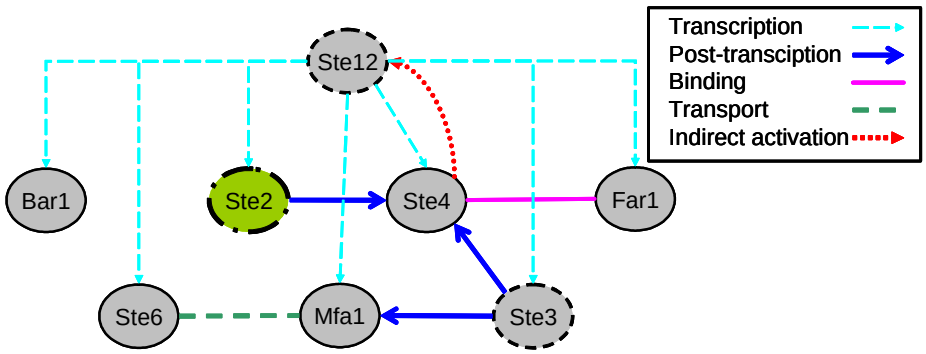

Figure 3.7: Molecular interactions associated with Ste2 hub compiled from the literature: Ste2 is activated by the alpha mating factor. It triggers a signalling pathway that activates Ste12, which in turn transcribes the genes in the hub. Furthermore, many of the hub genes are directly interact with each other. Genes with a solid border belong to the hub, while genes with a dashed border do not.

which haploid cells of opposite mating type synchronize their cell cycles so that they can fuse and become a diploid. In summary, all genes captured in the hub are closely involved with a response triggered by Ste2.

Above, we provided examples in which two *hub* genes, each at the top of its respective signalling cascade, explained away the dependence and correlation between their effector genes. In the case of Slt2, which activates a number of transcription factors, we captured a broad response. For Ste2, we reconstructed a very precise and focused response. This demonstrates the success of Bayesian networks in inferring both broad and specific responses.

Unfortunately, not all hubs are easy to interpret. For instance, Lys1, a protein involved in lysine biosyntheses, is a hub gene that interacts with 14 gene mate at confidence > 0.4[2]. The genes associated with Lys1 are involved in a wide variety of amino acid related processes: hisidine biosynthesis (His7, His3, His5), metionine metabolism (Hom3), lysine biosythasis (Lys20), and mitocondrial related (Ilv2, Yhm1). We have no explanation, neither for Lys1's central role, nor for the logic behind this particular mixture of metabolic genes. Such a metabolic hub could hold some yet to be discovered insight into the cross-talk between metabolic pathways or it could simply be an artifact of the computational method.

3.6 Subnetworks

In the previous section we saw that while gene mate and separator features provide us with important insights, hubs that bring together multiple relations offer a broader perspective. Combining features in concert provides two key advantages:

[2]Unlike the previous examples, this is not a perfect hub: 12 of the possible 81 gene mate relations exist between Lys1's mates.

1. A richer more structured context for exploring the molecular interactions represented by our network.
2. More importantly, by combining a multitude of features, we can gain a more confidence.

When examining relations between two or three genes we limited ourselves to confidence > 0.75 (see Section 3.7.1 for justification). This approach can be overly cautious, discarding correct features whose confidence is below our threshold. On the other hand, in our analysis of hubs, we found most relations with confidence as low as 0.4 to be biologically correct. These features, while not significant in isolation, gained confidence based on the other features around them.

In this section we extend the concept of a hub to more general graph structures. We hypothesize that if we can find a subnetwork that contains a "surprising concentration" of high confidence gene mates, then our estimate of features in this region will be more reliable. Thus, while a full-scale network is currently of insufficient quality, self-contained subnetworks can be reconstructed based on individual features of high confidence. Indeed, such subnetworks often correspond to a coherent biologically process or response.

3.6.1 Constructing Subnetworks

A *subnetwork* is a weighted graph representing a subset the gene mate relations. While this graph encodes structural relations between variables in $\mathcal{X}$, it no longer follows the formalism of Bayesian networks.

Definition 3.6.1: We define a *t-Markov Graph* of an ensemble of Bayesian networks, denoted MG_t as follows: The vertices correspond to the random variables $X_1, \ldots, X_n$. An edge exists between X_i and X_j iff they participate in a gene mate relation with confidence greater than t. The edge (i, j) is weighted with the confidence of the relationship between X_i and X_j. ∎

Our goal is to search for an induced sub-graph of MG_t which is "significant". We evaluate this significance in terms of the edge weights in the sub-graph. We define a measure of "surprise" for a sub-graph induced by subset of variables: Let $F(c)$ be the probability of randomly choosing an edge of weight $\geq c$ from MG_0. We estimate this probability by computing the observed fraction of gene mates with confidence $\geq c$ among the $\binom{n}{2}$ possible gene pairs in our domain. Consider a set of k variables, $\mathbf{U} \subset \mathcal{X}$, and the subgraph they induce in MG_t. This subgraph contains the edges $e_1, \ldots, e_l$ with weights $c_1 \geq c_2 \geq \cdots \geq c_l \geq t$, respectively. We evaluate the significance of this subgraph by bounding the expected number of subgraphs with similar weight or higher that we expect to find under a null-hypothesis model. Our null model assumes that the confidence of each edge is sampled independently from the distribution F.

The probability of sampling l edges with weights $c_1, \ldots, c_l$ or higher, is $\prod_{i=1}^{l} F(c_i)$. Given k variables, there are $\binom{K}{l}$ (where $K = \binom{k}{2}$) ways to choose l edges between the K possible pairs of variables. Thus, the probability of randomly sampling a subgraph with k variables having at

least l edges with weights better than $c_1, \ldots, c_l$ is at most $\binom{K}{l} \prod_i F(c_i)$. We search for a subgraph $H \subset MG_t$ over all possible subsets of size k in $\mathcal{X}$, thus the *expected number* of occurrences of such a subgraph is at most

$$S(H) = \binom{n}{k}\binom{K}{l}\prod_i F(c_i) \tag{3.2}$$

We use Eq. (3.2) as a scoring function to measure the significance of a subgraph. We call a high scoring subgraph a *subnetwork*. Given our scoring function, we devise an algorithm to find high-scoring subnetworks in MG_t. Our approach begins with a *seed*, a small set of variables $\mathbf{S} \subset \mathcal{X}$ and tries to expand the seed into a high scoring subnetwork. A seed can be a set of functionally related genes manually defined by a biologist. A more automated definition of a seed is a connected component with at least d variables in MG_{ts} where $ts > t$.

A radius based approach can be used to expand a seed by defining the subnetwork as all variables within a sphere of radius l around $\mathbf{S}$ in MG_t.

Definition 3.6.2: A *sphere radius of* l around $\mathbf{S}$ in MG_t is the set of all variables X for which $\exists s \in \mathbf{S}$ s.t. there is a path between X and s of lenght $< l$ in MG_t. ∎

The rational behind this approach is that genes in close proximity of the seed in MG_t are more likely to belong to the same process or pathway. We then score the resulting subnetwork using Eq. (3.2) and keep it if it scores better than a pre-specified threshold tE. In practice we use $tE = e^{-5}$, $t = 0.5$, $ts = 0.8$, $d = 3$, and $l = 2$.

A more systematic approach to search for high scoring subnetworks, would be by employing a greedy hill-climbing algorithm. This search starts with a candidate seed and at each step considers adding or removing a single variable. The algorithm selects the operation that leads to the largest increase in score. Once we reach a local optimum, we keep the subnetwork if its scores higher than our threshold tE. After systematically expanding all candidate seeds we can filter out subnetworks with high similarity and return a unique set of different subnetworks.

The hill-climbing approach has a number of advantages: It selectively includes only variables that increase the significance of the subnetwork. Thus, it can expand the network in an a-symmetrical manner, only in directions which increase the score. It also continues to expand the subnetwork until a local maximum is reached. Due to this increased sensitivity, we use the hill-climbing approach on $MG_{0.3}$ (verses $MG_{0.5}$ for the radius based approach).

While the hill-climbing approach finds higher scoring subnetworks, in some cases the radius based procedure achieves results with better biological coherence. When using a seed of genes annotated with the same biological process, a bounded radius is more likely to result in a subnetwork that remains related to that same process. When using the greedy approach, the subnetwork sometimes gets drawn into a higher scoring neighboring processes (in MG_t) which are not necessarily functionally related to the process represented by the seed. Thus, different insights can be gained by applying both the radius based and the greedy procedure in search of highly significant subnetworks.

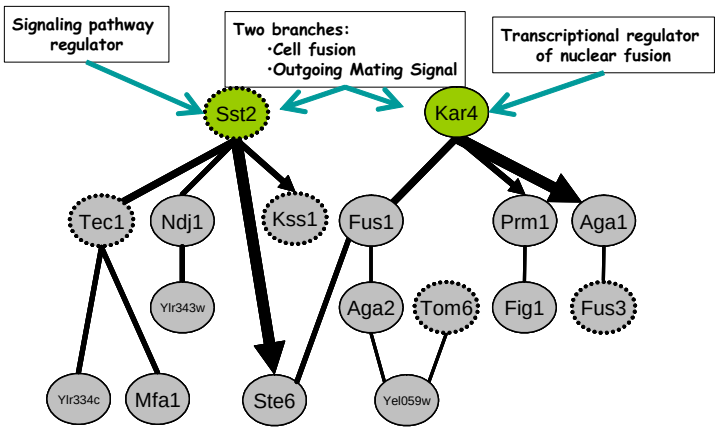

Figure 3.8: Mating subnetwork: The mating subnetwork reconstructed from variance-oriented dataset. The thickness of each edge in the network corresponds to the confidence of the gene mate relation. An edge is directed if it is directed in more than 50% of the PDAGs. Genes with a dotted border were mutated in some samples in the dataset.

3.6.2 Biological Subnetworks

The full power of our approach becomes apparent when exploring subnetworks. Applying the radius based approach to the variance-oriented dataset resulted in 6 well-structured subnetworks, each representing a coherent molecular response: mating response, low osmolarity cell wall integrity pathway, stationary phase response, iron homeostasis, amino acid metabolism along with mitochondrial function, and citrate metabolism. Of 87 top scoring gene mates, 61 relations appeared within one of these subnetworks. Our score based approach to constructing subnetworks produced 5 highly significant networks, capturing 4 of the 6 partially hand-crafted networks.

While Hughes et al [51] identify some of these responses (amino acid metabolism, iron homeostasis, and mating) using clustering, our reconstructed networks provide a much richer context for regulatory and functional analysis. We use the mating response subnetwork, shown in Figure 3.8, to demonstrate the power of our method to reconstruct a coherent biological tale and raise novel biological hypotheses. We discern two distinct branches, one for cell fusion and the other for outgoing mating signaling. In our network, the cell fusion response branch is mediated by Kar4 and includes several known cell membrane fusion genes (Fus1, Aga1, Aga2, Prm1 and Fig1) as well as two genes previously unassociated with this process (Tom6 and Yel059w). The multitude of high confidence relations strongly suggests a putative role to Kar4 in regulating not only nuclear fusion but also cell membrane fusion. Another branch in this network is directed from Sst2, a mating signaling pathway regulator. Since an Sst2 mutant has been incorporated in the compendium, we could determine edge direction and identify Sst2 as a prime regulator of several other genes (Tec1, Ste6, Mfa1, Kss1) previously shown to be transcriptionally regulated by the mating pathway.

Some puzzling discrepancies exist in our network. The first is the absence of Ste12, the main

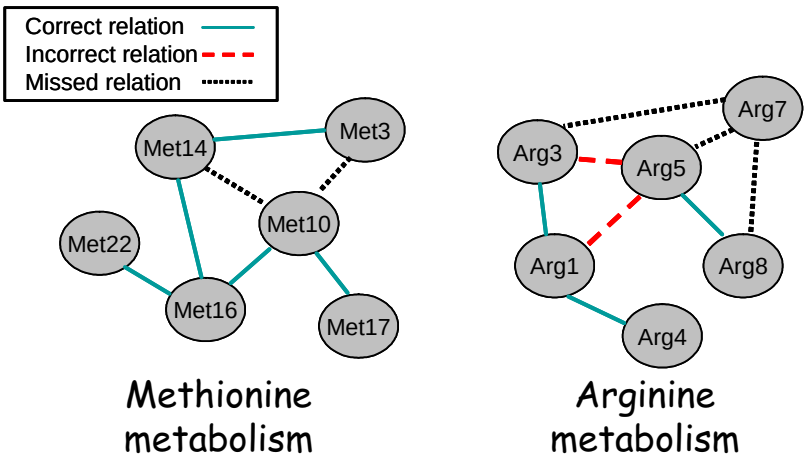

Figure 3.9: Amino Acid Metabolism fragments: AA metabolism fragments reconstructed from process-oriented dataset. Each fragment represents a connected component in $MG_{0.75}$.

transcription factor of the pathway. This is most likely due to loss of information by our discretization procedure. The second is the marginal position of the pathway's MAP kinase, Fus3. Despite the knockout mutation in Fus3 we have failed to identify directed regulation. We believe that a larger number of repetitions for each mutation will enhance our framework's capabilities to discover such regulatory relations.

The subnetworks resulting from the process-oriented produced a similar set of molecular responses: AA Metabolism, methionine and sulfur metabolism, lipids metabolism, phosphate metabolism, carbohydrate metabolism (especially glycogen), oxidate phosphorylation, iron metabolism, mating response, stationary phase response, and cell wall integrity response. One of the largest subnetworks is the amino acid metabolism subnetwork. While the entire subnetwork consists of a wide range of amino acid metabolism processes, we discern finer sub-components of higher confidence. Connections within each such component are of very high confidence, while the edges between components are of lower confidence. Figure 3.9 illustrates two such components: sulfate-methionine metabolism and uera cycle-arginine metabolism. Both components correspond very closely to the correct architecture of the metabolic pathway. Indeed, most edges represent metabolic steps, i.e. two genes that share a metabolite. We provide a systematic study on the accuracy of the Bayesian network reconstruction of metabolic pathways in Section 3.7.

In addition to systematically reconstructing molecular pathways, subnetworks can aid the functional annotation of uncharacterized genes. An example is the gene mate Pry2 – Svs1 (0.88,0.79). At the time of analysis, Pry2 was annotated as protein expressed under starvation and Svs1 was annotated as serine- and threonine-rich protein required for vanadate resistance. Therefore, this pair fell under the category of unclear/wrong gene mates (see Figure 3.4). A subnetwork of $MG_{0.7}$ around this pair is given in Figure 3.10. In this subnetwork we see that Pry2 and Svs1 appear in the midst of cell wall proteins and osmosensors suggesting a cell wall role, perhaps under osmotic

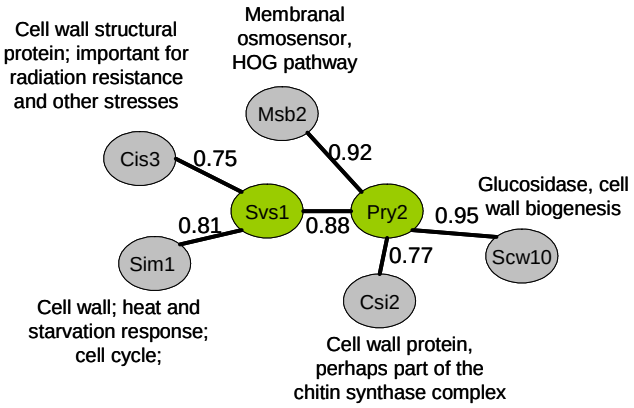

Figure 3.10: Functional annotation using Bayesian networks: A subnetwork around gene mates Svs1 and Pry2 from process-oriented dataset. This includes a ball of radius 1 around this pair in $MG_{0.75}$. Each edge is labeled with the confidence of the gene mate pair. Annotation from literature labels each of the surrounding genes.

stress. A recent paper [95], that did not exist at the time of our initial analysis, indicates that both Pry2 and Svs1 are indeed cell wall proteins. This confirms both our hypothesis and their coupling as gene-mates, suggesting that perhaps other unclear and unknown gene-mates represent correct relationships. Clustering is one of the standard methods used to assign functional annotation to uncharacterized genes. In the clustering of Hughes et al. [51]), while Pry2 and Svs1 do indeed fall into the same cluster along with Cis3 and Msb2, the cluster is not annotated as cell-wall. The entire cluster consists of a mixture of annotations dominated by cell-cycle and stress, and therefore does not facilitate the correct annotation of these genes as cell-wall proteins. Functional annotation based on the Bayesian network is potentially more reliable. We attribute this reliability to the high statistical significance of each relation included in the network and the finer context of these relations.

3.7 Systematic Evaluation

In Section 3.5 and Section 3.6 we gave many examples that demonstrated the ability of Bayesian networks to correctly reconstruct portions of molecular pathways. While including a large and comprehensive set of examples, with the exception of Figure 3.4, the analysis was anecdotal. In this section we analyze the statistical robustness of the reconstructed networks and systematically evaluate of the correspondence between the inferred relations and known biology. Together, these provide a more objective measure on the quality of Bayesian network analysis for gene expression data.

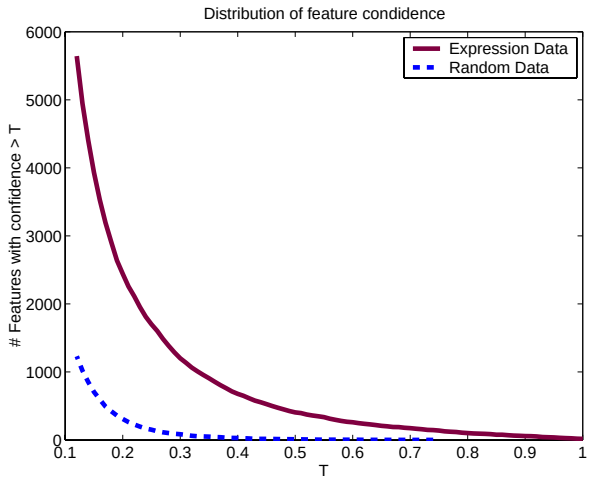

Figure 3.11: Comparison of feature confidence distribution between random and real data. The randomized dataset containing the highest feature confidence was chosen for this comparison.

3.7.1 Statistical Robustness

We performed a number of tests to examine the robustness and statistical significance of our learning procedure. One of the most important issues is: Which confidence levels represent reliable gene mate relations? We estimate which confidence levels are statistically significant using both *randomization studies*, where we permute the expression of genes across experiments and *simulation studies*, where we generate training sets from a known synthetic network and evaluate how accurately we reconstruct this network [34, 36].

To assess the number of spurious relations in our data, we ran the entire process on three randomized datasets. We created each random dataset by randomly permuting the order of the experiments independently for each gene. Thus, for each gene the order of arrays was random, but the composition of the series remained unchanged. In such a dataset genes are independent of each other and there are no "real" dependencies. The random data has no feature with confidence above 0.74, 0.61 and 0.44 respectively for each dataset. We compare the distribution of confidence estimates between the biological data set and the randomized set, Figure 3.11 plots this comparison. As expected, the gene mate relations in the random dataset have significantly lower confidence. This leads us to believe that features inferred from the biological dataset with confidence above 0.75 originate from true signals in the data. Note, while we have demonstated that gene mates represent true signals, this analysis does not provide any support for our interpretation of this signal.

A big concern regarding our approach is whether it is possible to correctly infer relations between so many variables given so few samples. To test this, we created a synthetic model. To simulate a distribution similar to that observed in the real data, we used one of the networks learned from the real data as our synthetic network. We sampled 300 random data instances from this synthetic network and applied our entire process to the resulting dataset. Since the generating network

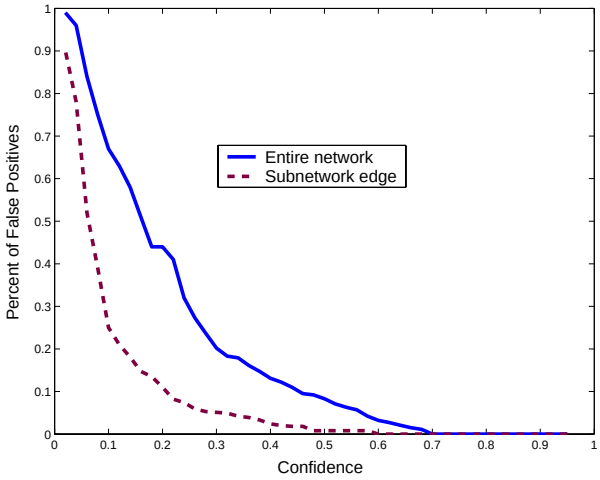

Figure 3.12: False positive rate in synthetic dataset

was known, we could accurately evaluate our false positive rate (see Figure 3.12). The graph shows that the false positive rate is close to 0 for confidence levels almost as low as 0.6. While the synthetic model represents a distribution similar to the empirical distribution, reconstruction of the synthetic model is a substantially easier task. Contrary to real data, the synthetic model contains discrete data (verses discritization of a continuous signal), no noise, no hidden variables and no feedback loops. Therefore, we take a more conservative threshold and along with the results from the randomization studies adopt confidence 0.75 as a reliable threshold for gene mate relations.

In addition to gene mate relations, we evaluated the statistical significance of subnetworks. We applied our greedy subnetwork procedure to features learned from the randomized dataset. Indeed, none of the resulting subgraphs scored above our threshold. Furthermore, we used our synthetic dataset to test if feature confidence within a subnetwork is more significant than the confidence of an isolated feature. An *subnetwork edge* is defined as a markov feature between two genes participating in the same significant ($S(H) < e^{-5}$) subnetwork. We believe that subnetworks correspond to a true signal in the data. Therefore, a subnetwork edge is more likely to be correct than an isolated edge. We computed the false positive rate of subnetwork edges (see Figure 3.12). Indeed, the false positive rate within the subnetworks is significantly lower, affording us assurance for features with confidence as low as 0.4.

Finally, since the analysis was not performed on the entire *S. cerevisiae* genome, we also tested the robustness of our analysis to the addition of more genes. We compared gene mates relations and their confidences resulting from the analysis of 2 datasets over the same samples, one set extending the gene set of the other. The first dataset contained 200 genes annotated with amino acid metabolism and related processes. The second dataset extended the first set with an additional 200 genes whose expression was highly correlated to the genes in the first set. Figure 3.13 compares feature confidence between the two datasets. The figure demonstrates a strong correlation between

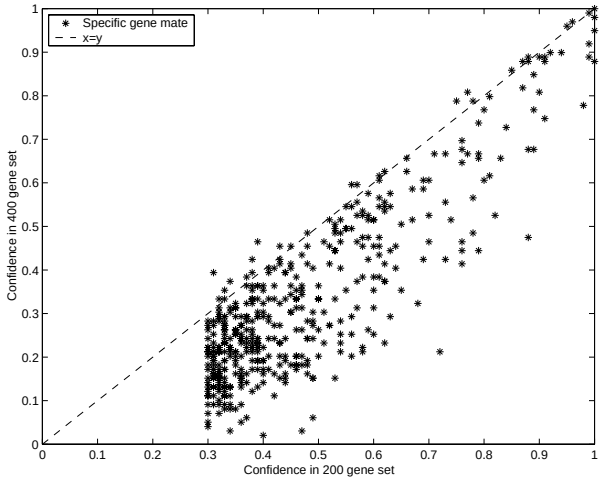

Figure 3.13: Comparison of feature confidence in the 200 gene set and the 400 gene set

feature confidence levels. As expected, the confidence in the larger dataset is usually lower due to new relations which form in the larger set: a gene from the small set finds a "better" mate in the larger set.

3.7.2 Comparison to Literature

While the statistical analysis presented in Section 3.7.1 provides some indication that our Bayesian network procedure might be capable of reconstructing molecular interactions, a more important question is "How closely does the automated reconstruction correspond to the true biological network?" Unfortunately, the true biological network is not known. Instead, we construct a partial picture of the biological ground truth based on the current biological literature and databases.

Our validation procedure focuses on two biological processes: the mating response (signalling) and amino acid (AA) metabolism. Both processes correspond to subnetworks automatically reconstructed from the variance-oriented dataset[3] for which there is an extensive knowledge base. We used information from SGD [11], YPD [21], and KEGG [59] along with published literature to hand curate a reference network for each process. First, we compiled a list of genes associated with each process, resulting in 99 mating genes and 185 amino acid metabolism genes[4]. We then compiled a comprehensive listing of all known interactions between these genes. We included a diverse set of molecular interactions: transcriptional regulation, post-transcription regulation, post-transcriptional signalling, protein-protein binding and metabolic links. We combined these interactions to create undirected reference networks for both mating and AA metabolism. We note that the resulting

[3]Whose genes were chosen solely based on the variability of their gene expression in the Compendium [51] experiments

[4]Since the full AA metabolism network is too large, genes were selected as an extension of the gene set in the AA subnetwork of the variance-oriented dataset by building plausible pathways around them (based on databases and literature) and adding the associated genes

networks significantly extend the classic textbook pathways of these processes. We believe these networks constitute the closest approximation of the biological truth available at the time of their construction. Therefore, we treated these reference networks as the ground truth and compared our automated reconstructions to them.

We use the gene mate features reconstructed from the process-oriented dataset for this comparison. The gene set contains all mating and AA metabolism genes in the reference networks, together with all genes whose expression correlates well with these genes. In addition, we included genes whose expression had large variation across samples. There are a number of criteria by which we compare to the reference network, each representing a different degree of detail. At the finest degree we ask: What fraction of the gene mates exactly correspond to an interaction in our reference network? This measures how many gene mates represent a true direct molecular interaction. We can relax this criteria and ask which gene mates appear "close" in the reference network? At a much coarser level we can ask what fraction of the gene mates are connected via a path in the reference network. Since the process-oriented dataset contains ≈ 1000 genes, even success in this coarse test is significant.

Initially, we evaluated our ability to correctly reconstruct direct molecular interactions. We limited our evaluation only to gene mates for which both genes belonged to the same pathway (our reference networks provided no information regarding other types of interactions). We calculated what fraction of these inferred interactions correspond to an edge in our reference network. In addition, we assigned a p-value to this fraction using the hyper-geometric distribution. In a nutshell, this measures the probability of achieving k successes (number of correct pairs) out of n trails (number of inferred gene mates) assuming that the genes in each pair were chosen uniformly at random. Given the parameters K (number of correct interactions in our network) and N (number of possible pairs), the probability of a single "success" is K/N. For the mating network, we tested all gene mates with confidence > 0.4 and found $3/21$ (pvalue 0.03) direct interactions. When applying the same procedure to a smaller dataset containing only the mating genes, we were more successful and captured $7/49$ (p 0.002) direct interactions[5]. Since there were many AA metabolism gene mates, we used a 0.5 confidence threshold. We found $10/46$ (p 2.0×10^{-6}) direct interactions of which 1 was regulatory and 9 were metabolic steps.

Next, we tested our inference at a coarser scale. We tested all pairs in which at least one gene belonged to the reference pathway (relaxing the requirement of both genes). A pair is marked a success if there is a path in the reference network that connects that pair. Even at this coarser level we attempted to capture structure that is finer than simple co-regulation. Since many genes in the pathway are co-regulated by transcription factors such as Ste12 and Gcn4, we limited a connecting path between genes to contain at most one regulatory relation. Using the same confidence thresholds as before: For the mating response we found $18/55$ (p 4.2×10^{-16}) connected genes mates. For AA metabolism we found $37/70$ (p 1.2×10^{-47}) connected gene mates.

This evaluation establishes that the correspondence between our reconstructed network and the

[5]Only one of the seven relations is regulatory.

"biological truth" is statistically significant. The quality of our inference is beyond anecdotal. We note the following observations:

While the ability of our method to reconstruct direct molecular interactions is statistically significant, our method is more successful at reconstructing closely related but not physically interacting genes. Direct interactions are missed for two central reasons: The noise in gene expression data along with the statistical difficulties involving a small sample size impedes the correct detection of direct physical interactions. This can be partially remedied by collecting larger datasets. In addition, many direct interactions are missed because the true activity of many genes is simply not observable in gene expression data and thus hidden from our analysis. This is an intrinsic problem of gene expression data.

Our method is fairly weak at capturing regulatory relationships. While our first naïve hope might have been that direct transcriptional relations exhibit the strongest dependencies in gene expression data, this does not seem to be the case. The activation of many transcription factors is post-translational and thus hidden in expression data. Furthermore, many transcription factors are expressed in relatively small quantities, making them more vulnerable to noise. Therefore, dependency of co-regulated genes is often stronger. Reconstruction of regulatory relations based on dependencies will perhaps be more successful when protein expression data becomes available.

Finally, the ability of a Bayesian network to capture metabolic pathways at such fine detail is intriguing. We speculate that close metabolic links have a stronger co-regulatory mechanism. This mechanism might not manifest itself in the transcriptional network itself and instead might be driven by other mechanisms. For instance, genes that share the same transcription factor might have different binding affinities in their promotors. The binding affinities of closely related genes might be more similar, leading to more finely tuned co-regulation, which can be detected by our analysis. Another mechanism that could be responsible for this tight co-regulation might be based on metabolites shared by these genes. This topic merits future research.

3.7.3 Comparison to Other Methods

In Section 3.7.2, we showed that our inferred gene mates correspond with known biological pathways in a statistically significant manner. But, this significance was measured relative to random pairs. In this section we demonstrate that our technique not only out performs choosing pairs at random, it is also superior to other analysis methods. We compare our method to three others: clustering, correlation and ranked correlation.

The reference networks described in Section 3.7.2 are used as the ground truth. A false positive represents a pair of genes which are not connected in the reference network. A true positive is a pair connected by a path of length at most l, we compare at two degrees of correspondence: $l = 2$ and $l = 6$. In this analysis we did not limit paths to contain only one regulatory relation and thus some of the pairs may represent co-regulation.

The question that arises is how can we compare the different methods? Each has an inherently different set of parameters. These parameters lead to different trade-offs between the number of

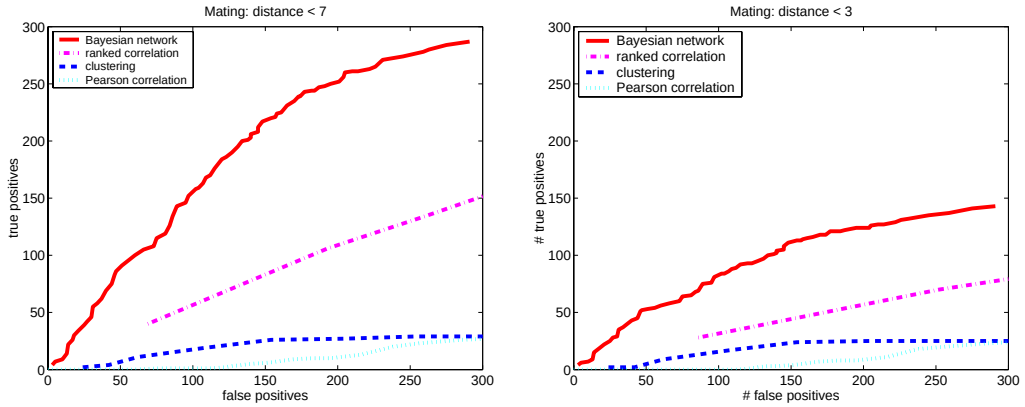

Figure 3.14: ROC curves for mating network

correct verses incorrect relationships. For example: At rank one, correlation contains 100 false positives and 25 true relations. At confidence 0.85, there are only 13 gene mates of which 4 are correct. Which is better? How one compare these results? A standard solution is to compare the methods across a wide range of thresholds. A plot of the resulting ratio between true positives and false positives at the different thresholds is termed a *ROC curve*. Thus, we can choose a specific number of true positives and compare the number of false positives between methods. If the curve for method A is consistently above the curve for method B, we can conclude that method A is better than method B.

The following describes how we defined thresholds for each of the methods:

- **Gene mates:** We test all pairs at confidence $> t$, incrementally lowering t.
- **Pearson correlation:** We test all pairs with absolute Pearson correlation $> t$, incrementally lowering t.
- **Ranked correlation:** For each gene, we calculate a ranked list of genes most strongly correlated with it. For a given reference gene, a gene of rank one means that no other gene in the data has higher correlation to the reference gene. Rank five means there are exactly four such genes. We rank a given pair with the lower of the two possible ranks. A pair can have low absolute correlation but low rank if there are no genes with higher correlation. A pair can have high rank but high absolute correlation if there are many other genes well correlated with each of the genes in the pair. We test all pairs with rank $< t$, incrementally raising t.
- **Clustering:** We clustered the genes using Eisen's hierarchical clustering algorithm [30]. This creates a hierarchy of clusters. For each depth in the hierarchy, we test all gene pairs that appear in the same cluster, incrementally decreasing the depth.

The resulting ROC curves for the mating pathway are presented in Figure 3.14 and for AA metabolism in Figure 3.15.

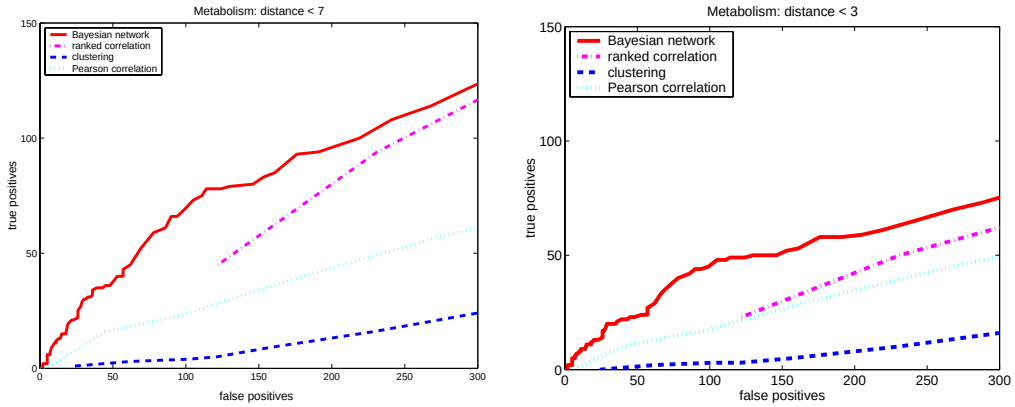

Figure 3.15: ROC curves for AA metabolism network

Before comparing the different methods, it is interesting to notice how the performance of the Bayesian network differs between the datasets. While AA metabolism was clearly superior to mating in the range of high confidence relations (see Section 3.7.2), the performance of mating significantly gains on AA metabolism for the middle and low confidence relations. We attribute the loss of performance in AA metabolism to two factors. First, unlike the mating pathway, our reference network for AA metabolism contains less than half the AA metabolism genes. Therefore, many true relations are missing in our network and labeled as false positives. We believe the performance would have improved had we constructed the full metabolic network. Second, AA metabolism is part of a very large cluster of highly correlated genes. The multitude of correlated genes along with the noise in gene expression data possibly obscures the correct reconstruction for all but the strongest signals.

All four ROC curves establish that Bayesian networks are substantially superior in reconstruction of molecular pathways. Ranked correlation is always the second best method. Notice, rank one already contains many false positives, while for Bayesian networks, thresholds with few false positives exist. For AA metabolism, pairwise correlation performed significantly better than clustering. This strengthens our observation that metabolic genes whom are closely related in the metabolic pathway are more strongly correlated in their expression than more distant metabolic genes. On the other hand, for the mating response, pairwise correlation was inferior to clustering. Indeed, most of the edges in the reference mating network have low correlation (average correlation of betweens genes sharing an edge in a mating network is very low). Therefore, correlation is less suited for reconstructing the mating response. We speculate that clustering (which is also based on correlation) gained robustness over pairwise correlation by requiring correlation between a set of genes over the same conditions.

Having demonstrated superiority of the Bayesian network approach, we ask: Which component of our method is responsible for its success?

One difference is that we used mutual information between discretized expression profiles,

verses correlation between the raw expression values used by the other methods. As noted in Section 3.5.1 about 1/3 of the gene mates have low correlation and are therefore less likely to be detected by correlation based methods. The key advantage to discretization is that it focuses on the conditions in which significant changes occurs. This concentrates most of the score on this set of significant conditions, while mostly ignoring spurious correlation under the typically large number of basal conditions in the data. On the other hand, discretization comes with a cost. While many correct gene mates are only detected due to the discretization, many other correct gene mates are lost due to bad discretization. We believe a more principled approach that automatically finds the correct trade off between discretization and raw signal can lead to improved results.

Another crucial factor leading to our success is the bootstrap analysis employed (see Figure 3.11 and Figure 3.12 for the relation between feature confidence and false positive rate). The importance of bootstrap is further supported by the inferior performance of the single highest scoring Bayesian network. Comparing to paths of length ≤ 2 in the reference network gives poor results: Mating - 40 true positives verses 206 false positives and AA metabolism- 61 true positives verses 354 false positives. Thus, without bootstrap, we have a high number of false positives and little confidence in our results.

Another key factor is our multivariate approach. Bayesian networks take a global viewpoint, evaluating sets of variables. In this approach, when one gene "explains" the behavior of another, other genes which provide a similar "explanation" become redundant. This feature is behind the sparsity of our network and leads to structures such as separators and hubs. One can view the ranked correlation as a simplistic version of the principle of explaining away. After a gene is "explained" by its d most highly correlated genes, no others are added. The relatively good performance of the ranked correlation method suggests that "explaining away" is an important factor in our success.

Unlike ranked correlation, a Bayesian network offers a principled approach to determining if a gene is redundant given another. How important is this principled approach? Testing the rank of all gene mates inferred with confidence > 0.75 reveals that most are rank one in respect to mutual information. Only 12 out of 138 gene mates (mostly metabolic) are an exception. This finding suggests that for this particular dataset, a method that applies bootstrap analysis to a simpler procedure based on ranking (either using mutual information or correlation) can lead to results of similar quality. We believe the capability of Bayesian networks to detect combinatorial regulation might come into play for more suitable datasets (e.g. more data, higher organisms or proteomic data).

3.8 Discussion

In this chapter we presented an approach for analyzing gene expression data that builds on the theory and algorithms for learning Bayesian networks. We described how to apply these techniques to gene expression data. The primary contribution of this approach is an automated methodology for finding detailed subnetworks of interacting genes. We demonstrated the use of this tool by analyzing

the Compendium data of *S. cerevisiae* mutations [51]. Both regulatory, metabolic, and signaling components are identified, showing the potential of our approach to uncover the three major types of molecular networks. In addition, we provide a systematic evaluation, demonstrating that the automatically reconstructed interactions do indeed correspond to known molecular pathways.

The biological motivation behind our approach was similar to previous work on inducing *genetic networks* from data [89, 1, 99]. At the time of publication, our method contributed two central novelties. First, the models we learn have probabilistic semantics. This better fits the stochastic nature of both the biological processes and the noisy experimental data. Second, our focus is on extracting features that are pronounced in the data, in contrast to previous genetic network approaches that attempt to find a single optimal model over all genes. To our knowledge, the global deterministic approaches were never successful on real data.

Our approach is quite different than clustering [3, 30, 51], it attempts to learn a much finer structure from the data. Our analysis illustrates the differences between our techniques and clustering methods. On the one hand, we are able to discover inter-cluster interactions between weakly correlated genes. On the other hand, we can uncover finer intra-cluster structure among correlated genes. This assists us in understanding the roles of genes within a richer context

While this thesis presents some encouraging results, reconstruction of molecular pathways from genome-wide data remains a difficult and unsolved problem. We list a number of drawbacks. First, our failure to reconstruct regulatory relations from gene expression data. In Chapter 5 and Chapter 6 we present extensions of Bayesian networks which are more suited towards reconstructing regulation from gene expression.

Another disappointment is that our analysis did not seem to utilize the full power of the Bayesian networks and could possibly have been achieved using much simpler add-hoc methods (see Section 3.7.3). Nevertheless, we feel that the full Bayesian network approach remains with merit. It provides a principled framework that can be further extended, tailored and developed towards the task of molecular pathway reconstruction. Many such developments have already been published: incorporating DNA binding data into the Bayesian network reconstruction [44], models that focus on reconstruction of regulation [77, 82], adapting Bayesian networks to E. coli time series data [72], or using more sophisticated local probability models [70]. We believe that many future successes in pathway reconstruction can and will be based on the foundations of Bayesian networks.

Chapter 4

Computational Methods for Learning Bayesian Networks

In this chapter we will present technical developments that solve two practical difficulties that arise when adapting Bayesian networks to the gene expression domain.

First, we present the *Sparse Candidate Algorithm*, an efficient heuristic search over Bayesian Network structures. In this thesis we address the optimization problem of finding a Bayesian network structure using a greedy hill climbing heuristic search, that is capable of handling the massive number of variables (genes) typically encountered when learning "genetic networks". When there are many variables, the search space is extremely large and standard search techniques can spend most of the time examining candidates that are extremely unreasonable. In Section 4.1 we introduce an algorithm that achieves quicker learning by restricting the search space to more likely candidates.

Second, we present a scoring function that takes genetic mutations into account. Some gene expression datasets [51, 53] include profiles of mutated genes. When a gene is mutated, it is disassociated from its natural regulatory mechanism. Instead, an external mechanism controls the expression of the gene. In Section 4.2 we present a modification to the Bayesian scoring function that deals with mutations.

4.1 The "Sparse Candidate" Algorithm

In this section we outline the framework for our *Sparse Candidate* algorithm [38]. In Section 2.3, we formulated learning structure of Bayesian networks as an optimization problem in the space of directed acyclic graphs. The number of such graphs is super-exponential in the number of variables. As we consider hundreds and possibly thousands of variables, the search space becomes prohibitively large. For such domains an efficient search technique is crucial.

We facilitate efficient learning by focusing the attention of the search procedure on more promising regions of the search space. In a nutshell, we identify a relatively small number of *candidate*

Input:

- A data set D,
- An initial network $\mathcal{G}_0$,
- A decomposable score $\mathrm{score}(\mathcal{G} : D) = \sum_i \mathrm{FamScore}(X_i, \mathbf{Pa}_i^{\mathcal{G}} : D)$
- A parameter k.

Output: A network structure $\mathcal{G}$.

Loop for $n = 1, 2, \ldots$ until convergence

Restrict: Based on D and $\mathcal{G}_{n-1}$, select for each variable X_i a set $\mathbf{C}_i^n$ ($|\mathbf{C}_i^n| \leq k$) of candidate parents. This defines a directed graph $H_n = (\mathcal{X}, E)$, where $E = \{X_j \rightarrow X_i | \forall i, j, X_j \in \mathbf{C}_i^n\}$. (Note that H_n is usually cyclic.)

Maximize: Find an acyclic network $\mathcal{G}_n$ maximizing $\mathrm{score}(\mathcal{G} : D)$ among networks that satisfy $\mathcal{G}_n \subset H_n$ (i.e., $\forall X_i, \mathbf{Pa}_i^{G_n} \subseteq \mathbf{C}_i^n$).

Return $\mathcal{G}_n$

Figure 4.1: Outline of the *Sparse Candidate* algorithm

parents for each gene based on simple local statistics (such as pairwise mutual information). We then **Restrict** our search to networks in which only these candidates of a variable can be its parents. This results in a much smaller search space in which we attempt **Maximize** the score of learned network. Our approach is iterative and uses the learned network to select better candidates for the next iteration.

4.1.1 Outline of Algorithm

Our procedure is comprised of two basic steps: In the **Restrict** step, we choose a set of variables $\mathbf{C}_X = \{Y_1, \ldots, Y_k\}$ that are the most promising *candidate* parents for X. In the **Maximize** step we search for a high scoring network constrained so that for every variable X we have $\mathbf{Pa}_X \subseteq \mathbf{C}_X$.

A possible pitfall of this approach is that once we choose the candidate parents for each variable, we are committed to them. Thus, a mistake in this initial stage can lead us astray into finding an inferior scoring network. We therefore iterate the basic procedure, using the constructed network to guide the selection of better candidate sets for the next iteration. The resulting procedure has the general form shown in Figure 4.1.

Before we go on to discuss the details of each step, we address the convergence properties of these iterations. Clearly, at this abstract level, we cannot say much about the performance of the algorithm. However, we can easily ensure its monotonic improvement. We require that in the **Restrict** step, the selected candidates for X_i's parents include X_i's current parents, i.e., the selection must satisfy $\mathbf{Pa}_i^{\mathcal{G}_n} \subseteq \mathbf{C}_i^{n+1}$ for all X_i. This requirement implies that $\mathcal{G}_n$, the network

chosen in the nth iteration, is a legal structure in the $n+1$ iteration. Thus, if the search procedure at the **Maximize** step also examines this structure, it must return a structure that scores at least as well as $\mathcal{G}_n$. Immediately, we get that $\text{score}(\mathcal{G}_{n+1} : D) \geq \text{score}(\mathcal{G}_n : D)$. Thus, the score improves with each iteration. In the following sections, we elaborate on each of the **Restrict** and **Maximize** steps.

4.1.2 Choosing Candidate Sets

Greedy hill-climbing search procedures for Bayesian network structure examine all possible local changes (e.g. edge addition, removal or reversal) and apply the one that leads to the biggest improvement in score. There are $O(n^2)$ possible changes, where n is the number of variables. It seems, however, that most of the candidates considered during the search can be eliminated in advance. For example, if X and Y are almost independent in the data, we might decide not to consider Y as a parent of X[1]. More generally, we use statistical cues from the data to restrict the set of networks we are willing to consider. We use measures of dependency between pairs of variables to *focus* our attention during the search. For each variable X, we find a set of variables, $Y_1, \ldots, Y_k$, that are the most promising *candidate* parents for X. The intuition is that if there is a strong dependency between two variables X and Y, then they should be "near" each other in the network. Thus, instead of having $n-1$ potential parents for a variable, we only consider k possible parents, where $k \ll n$.

The general outline of the **restrict** step is shown in Figure 4.2. We assign each X_j some score of relevance to X_i. As the candidate set of X_i, we choose those variables with the highest score. One could consider several possible scores to measure relevance of a potential parent X_j to X_i. In Friedman et al. [38] we examine alternative scores and compare their performance. Based on our experiments, one of the most successful measures, denoted $\mathrm{M}_{\text{Score}}(X_i, X_j : \mathcal{G})$, is simply the improvement X_i's local score if we were to add X_j as an additional parent. In the following, we discuss the rationale behind this score, by reviewing weaknesses of simpler scores that lead us to using $\mathrm{M}_{\text{Score}}(, :)$.

We first consider a simple and natural measure of dependence, which is *mutual information*:

$$\boldsymbol{I}(X;Y) = \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)} \tag{4.1}$$

Where $\hat{P}$ denotes the observed frequencies in the dataset. The mutual information is always non-negative. It is equal to 0 when X and Y are independent. The higher the mutual information, the stronger the dependence between X and Y (see [22] for more details).

While in many cases mutual information is a good criterion for the candidate parents, there are simple cases for which this measure fails.

[1] This is a heuristic argument, since X and Y can be marginally independent, yet have strong dependence in the presence of another variable (e.g., X is the XOR of Y and Z). We assume such cases are rare.

Input:

- Data set D
- A network $\mathcal{G}$,
- parameter k.

Output: For each variable X_i, a set of candidate parents $\mathbf{C}_i$ of size k.

Loop for each X_i $i = 1, \ldots, n$

- Calculate $\mathrm{M}_{\mathrm{Score}}(X_i, X_j : \mathcal{G})$ for all $X_j \neq X_i$ such that $X_j \notin \mathbf{Pa}_i^{\mathcal{G}}$
- Choose $x_1, \ldots, x_{k-l}$ with highest ranking , where $l = |\mathbf{Pa}_i^{\mathcal{G}}|$.
- Set $\mathbf{C}_i = \mathbf{Pa}_i^{\mathcal{G}} \cup \{x_1, \ldots, x_{k-l}\}$

Return $\{\mathbf{C}_i\}$

Figure 4.2: Outline of the *Restrict* step

]t

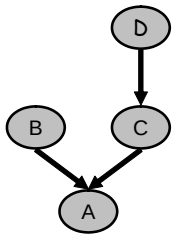

Figure 4.3: Network for Example 4.1.1

Example 4.1.1: Consider a network with 4 variables $A, B, C,$ and D illustrated in Figure 4.3. We can easily select parameters for this network such that $\boldsymbol{I}(A; C) > \boldsymbol{I}(A; D) > \boldsymbol{I}(A; B)$. If we select only two candidates for A, based on mutual information, we would select C and D. These two, however, are redundant since once we know C, D adds no new information about A. Moreover, this choice does not take into account the effect of B on A, and would therefore fail to find this true parent of A. ∎

This example shows a general problem in pairwise selection, which our iterative algorithm attempts to overcome. After we select C and D as candidates, a good learning procedure only sets C as a parent of A. Given that C is a parent of A, we must reestimate the relevance of B and D.

Which score can correctly reestimate the relevance? Recall that in a Bayesian network, X_i's parents *shield* it from its non-descendants (see Section 2.1.2). This suggests that we measure whether the conditional independence statement "X_i is independent of X_j given $\mathbf{Pa}_i$" holds. Equivalently, we can estimate how strongly this statement is violated. The natural extension of mutual information

for this task, is the notion of *conditional mutual information*:

$$\boldsymbol{I}(X;Y \mid Z) = \sum_{Z} \hat{P}(Z) \sum_{X,Y} \hat{P}(X,Y|Z) \log \frac{\hat{P}(X,Y|Z)}{\hat{P}(X|Z)\hat{P}(Y|Z)} \tag{4.2}$$

This measures the error we introduce by assuming that X and Y are independent given different values of Z. When $\mathcal{G}$ is the empty network, this measure is equivalent to $\boldsymbol{I}(X;Y)$. Note that although shielding can remove X's ancestors from the candidate set, it does not "shield" X from its descendants.

A deficiency of this measure is that it does not take into account the cardinality of various variables. For example if both Y and Z are possible candidate parents of X, but Y has two values (one bit of information), while Z has eight values (three bits of information), we would expect that Y is less informative about X than Z. On the other hand, we can estimate $P(X|Y)$ more robustly than $P(X|Z)$ since it involves fewer parameters.

Such considerations lead us to define a score which penalizes structures with more parameters: The more complex the model is, the easier we are misled by the empirical distribution. In order to design such a score, we reexamine the shielding property. Using the chain rule of mutual information:

$$\boldsymbol{I}(X_i; X_j \mid \mathbf{Pa}_i) = \boldsymbol{I}(X_i; X_j, \mathbf{Pa}_i) - \boldsymbol{I}(X_i; \mathbf{Pa}_i) \tag{4.3}$$

That is, the conditional mutual information is the additional information we get by predicting X_i using X_j and $\mathbf{Pa}_i$, compared to our prediction using $\mathbf{Pa}_i$ alone. Since the term $I(X_i; \mathbf{Pa}_i)$ does not depend on X_j, we don't need to compute it when we compare the information that different X_j's provide about X_i. Thus, $\boldsymbol{I}(X_i; X_j, \mathbf{Pa}_i)$ is an equivalent measure to conditional mutual information.

Now, if we consider the score used in learning Bayesian network structure (see Section 2.3.2) as cautious approximation of the mutual information with a penalty on the number of parameters, we get the *score* measure;

$$\mathrm{M}_{\mathrm{Score}}(X_i, X_j : \mathcal{G}) = \mathrm{FamScore}_{\mathcal{B}}(X_i, X_j \cup \mathbf{Pa}_i \ : \ D) \tag{4.4}$$

This simply measures the score when adding X_j to the current parents of X_i. This completes the motivation behind the derived score.

4.1.3 Learning with Small Candidate Sets

In this section we examine the problem of finding a constrained Bayesian network attaining a maximal score. Formally, we attempt to solve the following problem:

Definition 4.1.2: Maximal Restricted Bayesian Network (MRBN)
Input:

- A set D of instances

- A directed graph H of bounded in-degree k
- A decomposable score S

Output: A network structure $\mathcal{G}$ so that $\mathcal{G} \subseteq H$, that maximizes S with respect to D.∎

Proposition 4.1.3: *MRBN is NP-hard.*

This follows from a slight modification of the NP-hardness of finding an optimal unconstrained Bayesian network [12]. Therefore, we resort to using the greedy hill-climbing algorithm described in Section 2.3.2. The only difference being that we restrict the "Add" operator to the candidate parent sets.

If MRBN remains computationally hard, what do we gain by using the sparse candidate algorithm? Our gain is two-fold: First, when limiting the possible parents of each variable, the search space becomes considerably smaller, thus with the same number of steps we sample a larger portion of the space. Furthermore, the computational cost of each such step is significantly reduced. We elaborate on these points.

The number of possible Bayesian networks is extremely large. Assume we limit each variable to at most k parents, then there are $O(\binom{n}{k})$ possible parent sets. If the choice of parents for each variable were independent, there would be $O(\binom{n}{k}^n)$ possible networks to search over. Of course, acyclicity constraints disallow many of these networks, but removing cyclic networks from consideration does not have a significant effect on the number of networks. On the other hand, in MRBN, we have only $O(2^k)$ possible parent sets for each variable. Thus, while the search space for MRBN remains exponential in size, it is an order of a magnitude smaller than the search space for Bayesian networks with a bounded in degree.

Examining the time complexity for each iteration also points in favor of MRBN. When unconstrained, the initial iteration of greedy hill climbing considers all possible directed edges between any pair of variables in $\mathcal{X}$, calculating the score for each of these $O(n^2)$ initial edge additions to the network. After that, each iteration requires calculation all $O(n)$ possible edge additions, removals and reversals for the variables whose family changed in the previous iteration. In comparison, for MRBN we begin with $O(kn)$ initial calculations after which each iteration only requires $O(k)$ calculations.

Finally, A large fraction of the learning time involves collecting sufficient statistics from the data. Such statistics are needed for each combination of variable and parents considered during the learning. We gather these counts from the input data instances, often reading them over many times. Here again restricting to candidate sets saves time. When k is reasonably small, we can compute the statistics for $\{X_i\} \cup \mathbf{C}_i$ in one single pass over the input. All the statistics we need for evaluating subsets of $\mathbf{C}_i$ as parents of X_i can then be computed by marginalization from these counts. Thus, we can dramatically reduce the number of statistics collected from the data.

A Divide and Conquer Paradigm

In this section we show how can the combinatorial properties of the candidate graph H be utilized in order to efficiently find the maximal scoring network. Evidently, what makes MRBN hard is the acyclicity constraint. Otherwise, we would have selected, for each variable X_i, the parents that attain maximal weight.

"Divide and Conquer" is one of the most effective paradigms for designing algorithms. We apply this paradigm to MRBN as follows: first, we decompose the graph into components, so that their solutions efficiently combine in a acyclic manner. Next, given such a decomposition, we find acyclic solutions in each component and combine them into a global solution. This reveals another key advantage of MRBN over regular Bayesian network optimization, which is that the sparse nature of the candidate graph H often allows efficient decompositions of this sort.

The simplest such decomposition of this form is one that disallows intercomponent cycles, i.e, *strongly connected components*.

Definition 4.1.4: [20] Given a directed graph $H = (V, E)$, a subset of vertices $\mathbf{A} \subseteq V$ is *strongly connected* if for each $X, Y \in \mathbf{A}$, H contains a directed path from X to Y and a directed path from Y to X. The set $\mathbf{A}$ is *maximal* if there is no strongly connected superset of $\mathbf{A}$.∎

It is clear that two maximal strongly connected components must be disjoint, and there cannot be a cycle that involves vertices in both of them (for otherwise their union would be a strongly connected component). Thus, there is a unique partition of the vertices in H into maximal strongly connected components. Every cycle in H will be contained within a single component. Thus, once we ensure acyclicity of a chosen solution subgraph "locally" within each component, we get an acyclic solution over all the variables. This means we can search for a maximum on each component independently.

In order to decompose strongly connected graphs, we must consider potential cycles between the components. Therefore, our goal is to find small "bottlenecks" through which these cycles must go. We then consider all possible ways of breaking the cycles at these bottlenecks. The *separators* defined below are such bottlenecks.

Definition 4.1.5: A *separator*[2] of a directed graph $H = (V, E)$ is a set $S \subseteq V$ of vertices so that:

1. $H \setminus S$ has two components H'_1 and H'_2 with no edges between them. For $j \in \{1, 2\}$ let $H_i^S = H'_i \cup S$. (We omit the superscript where it is clear from context)
2. For each X_i,$\exists j \in \{1, 2\}$ so that $\{X_i \cup \mathbf{C}_i\} \subseteq H_j$

∎

[2]The standard definition of an undirected graph seperator requires only the first of the two items in the definition. The second item is specific to our use of the term

The second property ensures that, for each variable, we can search for the optimal choice of parents in only one component (H_1 or H_2). In the divide and conquer algorithm for MRBN we search for two optimal acyclic solutions $G_1 \subseteq H_1$ and $G_2 \subseteq H_2$ independently. Unfortunately, the combined graph $G = G_1 \cup G_2$ might be cyclic. The first property of separators ensures that the source of potential cycles in G involve at least two vertices in the separator S.

This suggests a way of ensuring that the combined graph will be acyclic. If we force some order on the vertices in S and require both G_1 and G_2 to respect this order, then no cycles can form. Given a small separator S, our approach considers all $|S|!$ possible orders. For each order, we independently find the optimal G_1 and G_2 that respect this order. We finally choose the order that maximizes the score of $G = G_1 \cup G_2$.

We can recursively decompose H using separators. A *cluster tree*, defined below, is a representation of such a recursive decomposition. The idea is similar to those of standard clique-tree algorithms used for Bayesian network inference (e.g., [57]).

Definition 4.1.6: A *Cluster Tree* of H is a pair $(\mathbf{U}, T)$, where $T = (J, F)$ is a tree and $\mathbf{U} = \{U_j | j \in J\}$ is a family of *clusters*, subsets of $\{X_1, \ldots, X_n\}$, one for each vertex of T, so that:

- For each X_i, there exists $j \in J$ such that $\{X_i \cup C_i\} \subseteq U_j$.
- For all $i, j, k \in J$, if j is on the path from i to k in T, then $U_i \cap U_k \subset U_j$. This is called the *running intersection property*.

∎

Definition 4.1.7: [8] The *treewidth* of a graph H is the size of the largest cluster (in number of vertices) in the cluster tree of H. ∎

In [38] we present a recursive dynamic program that uses cluster tree decomposition to search for the optimal Bayesian network constrained by H. Overall, the algorithm has the following complexity:

Theorem 4.1.8: Given a candidate graph H with a bounded indegree of k (number of candidates for each variable). If c is the treewidth of H and l is the number of clusters, then MRBN can be solved in $O(2^k \cdot (c+1)! \cdot l)$.

In summary, the algorithm is *linear* in the size of the cluster tree but worse than exponential in its treewidth. Furthermore, if there is a small (i.e. size of largest cluster is bounded by a constant) cluster tree, then it can be found in polynomial time [8]. Thus, MRBN has a polynomial time solution for all graphs of bounded treewidth.

4.1.4 Empirical Results

In this section, we validate the effectiveness of the Sparse Candidate algorithm by comparing it to standard greedy hill climbing (with no restriction to candidate parents). We compare these search procedures based on both their performance in the task at hand, and their computational cost.

For this comparison, we devised synthetic dataset. We used the cell cycle expression data of Spellman et al. [90]. This data set contains 76 gene expression profiles that measure six time series each using a different cell cycle synchronization method. Spellman et al. [90] identified 800 genes whose expression varied during cell-cycle. We learned a Bayesian network over these 800 genes and then sampled 5000 instances from the learned network.

In the reported experiments we use this same implementation of the greedy hill-climbing procedure both for the Maximize phase of the sparse candidate algorithm, and as a search procedure by itself. Thus the only difference is the restriction of the local changes to the candidate parent set. In all of our experiments we use the BDe score of [46] with a uniform prior. At each iteration, the procedure examines the change in the score for each possible move, and applies the one that leads to the biggest improvement. These iterations are repeated until convergence. In order to escape local maxima, the procedure is augmented with a simple version of TABU search. It keeps a list of the N last candidates seen, and instead of applying the best local change, it applies the best local change that results in a structure not on the list. Note that because of the TABU list, the best allowed change might actually reduce the score of the current candidate. We terminate the procedure after some fixed number of changes failed to result in an improvement over the best score seen so far. After termination, the procedure returns the best scoring structure it encountered.

We evaluate the quality of the networks found by each algorithm based on their score. The cost is evaluated based on both the running time and the number of sufficient statistics computed from the data. We report running times on a Pentium II 300MHz machine running Linux. To minimize the number of passes over the data we use a cache that allows us to use previously computed statistics and to marginalize them for obtaining the statistics of subsets. We report the number of actual de novo computation of statistics that were required by the algorithms.

The results are reported in figure 4.4. First, we note that while the first iteration of the algorithm finds reasonably high scoring networks, subsequent iterations improve the score. Thus, the re-evaluation of candidate sets based on our score does lead to important improvements. As we can see, the Sparse Candidate algorithm achieved significantly higher scores in much less time. The greedy hill-climbing search stopped because of lack of memory to store the collected statistics. At that stage it was far from the range of scores achieved by Sparse Candidate algorithm.

4.2 Modeling Mutations

In this section we present a scoring function that is suited for expression profiles of mutated strains and we discuss how gene expression profiles of mutated strains enhance learning causality. To

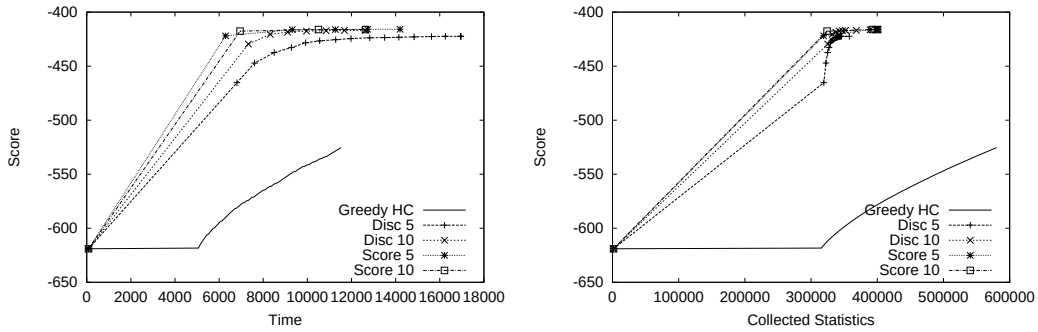

Figure 4.4: Comparison of the Sparse Candidate Algorithm with standard greedy hill-climbing. The greedy Hill-Climbing curve is significantly inferior to all the SPC curves. Score 5 and 10 represent the Sparse Candidate algorithm as presented in Section 4.1.2 with 5 and 10 candidate parents, respectively. Disc represents a slightly different score for re-estimating the candidates in each iteration.

develop this scoring function, we treat our Bayesian network as a *causal network*. Causal networks model not only the underlying probability distribution, but also the causal mechanism responsible for the observed behavior. Causal networks have a stricter interpretation w. r .t the meaning of edges: the parents of a variable are its *immediate causes*. The main difference between causal and Bayesian networks comes to play when we try to predict the outcome of an interventional query: What would happen if we mutate gene X?

4.2.1 Modeling an Intervention

One of the uses of microarrys is to measure the affect of a genetic mutation. In such an experiment, a gene (or set of genes) is either deleted or over-expressed and the global effect of this mutation on the expression of all genes is measured. Typical analysis [49, 51] takes a pairwise approach and tests the expression of which genes displays a significant change between the wild-type and the mutated strains. The list of differentially expressed genes is then inferred to be affected by the mutated gene, creating a list of relations: mutated gene X affects the expression of gene Y. In the following, we describe a principled method to incorporate expression profiles of mutated strains into learning of causal networks. Our approach leads to the inference of causal relations in a more global context.

Samples of mutant strains contradict a basic assumption made by our Bayesian network learning algorithm (Section 2.3), which is that each data instance was sampled from the same underlying distribution. For instance, by knocking out gene X, we replace the original molecular control on X's expression (its regulating parents) by an external one. Thus, any consequent measurement (in which X's value is constantly set to 0), will behave differently than X's conditional distribution on its parents in the wild type strains. Therefore, it is important to explicitly model this mutation into our learning algorithms.

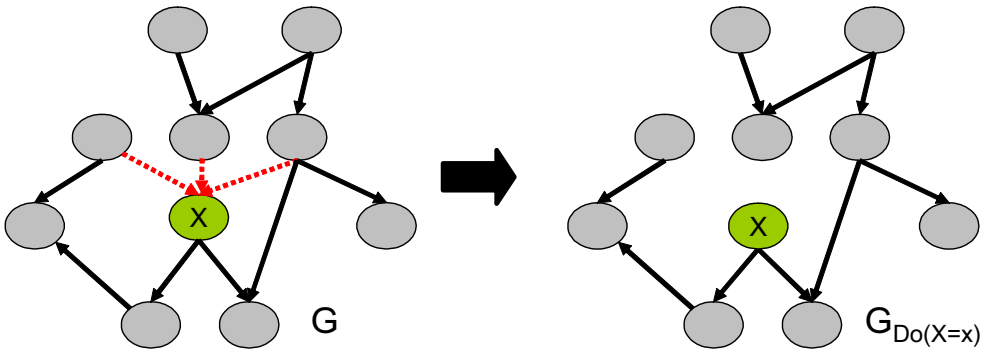

Figure 4.5: Example of ideal intervention: Assume the variable X is mutated in $\mathcal{G}$. We construct the corresponding $\mathcal{G}_{do(X=x)}$ by removing all edges incoming from $\mathbf{Pa}_X$ (red edges)

Formally, we model a mutation, denoted *do*(X=x), as an *Ideal Intervention* [74] which deterministically sets the expression of gene X to value x. This intervention disables the natural causal mechanisms that affect X and replaces them with an external deterministic mechanism. In addition, we assume that the intervention only affects X's causal mechanism and leaves intact all other causal mechanisms in the model, i.e., all other variables behave according to their respective conditional distribution. More formally, given a causal network $\mathcal{G}$, an ideal intervention defines a new causal network $\mathcal{G}_{do(X=x)}$, identical to the $\mathcal{G}$, except that all incoming edges into X are removed. In $\mathcal{G}_{do(X=x)}$, X becomes a root node associated with the probability distribution $Pr(X = x) = 1$ (See Figure 4.5 as an example). Note that X's outgoing edges are not affected by such an intervention. When a number of different variables are mutated in the same sample, we remove the edges incoming to each of these variables.

By modeling mutations, they can be used for causal inference. We illustrate this point with the following thought experiment. Consider the following pair of networks: $X \rightarrow Y$ and $Y \rightarrow X$. As Bayesian networks, the two are equivalent and can not be distinguished based on observational data alone (without mutations). Assume that we mutate X ($do(X = 0)$), then as causal models, we expect each of the two models to respond differently. If the causal model is $X \rightarrow Y$, then Y's causal mechanism remains intact. Therefore, the same conditional distribution measured in both the mutated and wild-type samples ($P(Y \mid do(X = x)) = P(Y \mid X = x)$). On the other hand, if $Y \rightarrow X$ is the causal model, the mutation disables the causal mechanism responsible for the dependency between the two variables. When X is mutated the variables X and Y become independent ($P(Y \mid do(X = x)) = P(Y)$). Therefore, we expect to measure a different conditional distribution in the mutant and wild-type samples. While we could not distinguish between the two models using observations (wild-type samples) alone, a mutation differentiates between them.

Note, in typical analysis of mutant strains the differentially expressed genes are inferred to be the *downstream* targets of the mutated gene. In contrast, the global nature of our reasoning allows us to reach a causal conclusions *upstream* of the mutation. In the above example, if we mutate X and observe a different conditional distribution between the wild-type and mutated samples ($P(Y \mid$

$do(X = x)) \neq P(Y \mid X = x)$), we infer that the causal mechanism between X and Y has been disrupted. Since a mutation in X disrupts the mechanism of its direct incoming causes, we infer that Y causes X, or in our terminology, Y regulates X.

4.2.2 Scoring with Mutations

The scoring function described in Section 2.3.2 assumes that all samples are drawn from the same network structure. We adapt that score to properly handle samples from different mutant strains (under the model of an ideal intervention). In such datasets, the underlying Bayesian network associated with each sample differs based on the mutated genes in the sampled strain.

Similarly to [17], we make the following set of assumptions:

- Our samples are independent random samples from a causal network. This causal network represents both the probability distribution sampled and the causal relationships in the data.
- Each mutation is an ideal intervention, i.e., it disables the normal causal mechanism of the mutated gene and an independent causal mechanism deterministically sets its value. Furthermore, this intervention does not directly affect the causal mechanism of any other gene.
- The data is complete, there are no missing or hidden variables.
- Global and local parameter independence (see Definition 2.3.4 and Definition 2.3.5).
- Parameter modularity (see Definition 2.3.7).
- The prior distribution of all parameters is Dirichlet.

One of the key properties of the Bayesian scoring function is that the score decomposes into a product of local entities, each depends only on X and $\mathbf{Pa}_X$ (see Proposition 2.3.8). Unfortunately, when D contains mutations, the instances are not associated with a single structure. Therefore, at first glance, such a decomposition might seem problematic. We shall now show how this problem can be overcome.

We define the following notations. Denote $\mathcal{M}[m]$ to be the set of mutations occurring in the mth sample. Let $\mathcal{M}$ be the collection of all sets of mutations occurring in $\mathcal{D}$ (We use the notation $\mathcal{M}[\mathcal{D}]$ when the dataset is not clear from the context). We donate by $\mathcal{G}^{\mathcal{M}}$ the set of all graphs derived from $\mathcal{G}$ based on the mutations occurring in $\mathcal{M}$, i.e., $\mathcal{G}^{\mathcal{M}} = \{\mathcal{G}_{do(\mathcal{M}[m])}\}_{m=1}^{M}$. Following the Bayesian approach, the appropriate *intervention score* would be:

$$\begin{aligned} \text{score}_{\text{Int}}(\mathcal{G} : \mathcal{D}) &= \log P(\mathcal{D} \mid \mathcal{G}, \mathcal{M}) P(\mathcal{G}) \\ &= \log P(\mathcal{G}) \int P(\mathcal{D} \mid \mathcal{G}, \mathcal{M}, \theta) P(\theta \mid \mathcal{M}, \mathcal{G}) d\theta \end{aligned}$$

Given that the instances are independently sampled from some causal model (some instances being observations and others ideal interventions), we decompose the likelihood by samples. The

probability of each sample is calculated using the appropriate graph, $\mathcal{G}_{do(\mathcal{M}[m])}$. Given the assumption of complete data, the likelihood decomposes further as follows.

$$\begin{aligned} P(\mathcal{D} \mid \mathcal{G}, \mathcal{M}) &= \log \int P(\theta \mid \mathcal{M}[m], \mathcal{G}) \prod_{m=1}^{M} P(\mathcal{X}[m] \mid \mathcal{M}[m], \mathcal{G}_{do(\mathcal{M}[m])}, \theta) d\theta \\ &= \log \int P(\theta \mid \mathcal{M}[m], \mathcal{G}) \prod_{m=1}^{M} \prod_{i=1}^{n} P(X_i[m] \mid \mathcal{M}[m], \mathbf{Pa}_i^{\mathcal{G}_{do(\mathcal{M}[m])}}, \theta) d\theta \end{aligned}$$

Our assumptions of parameter independence and parameter modularity allow us to calculate the contribution of each variable independently with out being affected by the choice of parent sets for other variables. Thus, the expression for each variable X_i decomposes as a separate integral that depends only on its parents in $\mathcal{G}_{do(\mathcal{M}[m])}$ and the associated parameters. The key point to notice is that while $\mathcal{G}^{\mathcal{M}}$ might contain many different graphs, if we focus our attention to a single variable, X_i, we find only two possible sets of parents: $\mathbf{Pa}_i^{\mathcal{G}}$ for samples where X_i in not mutated and X_i is a root in samples where it is mutated. Furthermore, the parameters of X_i in the non-mutated samples are independent of the mutated samples.

$$\begin{aligned} P(\mathcal{D} \mid \mathcal{G}, \mathcal{M}) &= \prod_{i=1}^{n} \int P(\theta_i \mid \mathcal{M}[m], \mathbf{Pa}_i^{\mathcal{G}}) \prod_{m=1}^{M} P(X_i[m] \mid \mathcal{M}[m], \mathbf{Pa}_i^{\mathcal{G}_{do(\mathcal{M}[m])}}, \theta_i) d\theta \\ &= \prod_{i=0}^{n} \int P(\theta_i \mid \mathcal{M}[m], \mathbf{Pa}_i^{\mathcal{G}}) \prod_{\mathcal{M}_i^0} P(X_i[m] \mid \mathcal{M}[m], \mathbf{Pa}_i^{\mathcal{G}}, \theta_i) d\theta \\ &\quad \int P(\theta_i \mid \mathcal{M}[m]) \prod_{\mathcal{M}_i^1} P(X_i[m] \mid \mathcal{M}[m], \emptyset, \theta_i) d\theta \end{aligned}$$

Where $\mathcal{M}_i^1 = \{m \mid X_i \in \mathcal{M}[m]\}$ and $\mathcal{M}_i^0 = \{m \mid X_i \notin \mathcal{M}[m]\}$. Notice that in the samples where $X_i \in \mathcal{M}[m]$, the value of X_i is deterministically set to x. Therefore:

$$\int P(\theta_i \mid \mathcal{M}[m]) \prod_{\mathcal{M}_i^1} P(X_i[m] \mid \mathcal{M}[m], \emptyset, \theta_i) d\theta_i = 1$$

According to our assumptions, for each variable X_i, both the likelihood and structure prior depends only on $\mathbf{Pa}_{\mathcal{G}}^i$ and θ_i. Thus, similarly to the case of observational samples alone, the scoring function decomposes into local functions associated with each variable. We therefore define:

$$\text{score}_{\text{Int}}(\mathcal{G} : \mathcal{D}) = \sum_{i=1}^{n} \log P(\mathbf{Pa}_{\mathcal{G}}^i) \int P(\theta_i \mid \mathcal{M}[m], \mathbf{Pa}_i^{\mathcal{G}}) \prod_{\mathcal{M}_i^0} P(X_i[m] \mid \mathcal{M}[m], \mathbf{Pa}_i^{\mathcal{G}}, \theta_i) d\theta \tag{4.5}$$

and denote the individual contribution of X_i's family by: $\mathrm{FamScore}_{\mathrm{Int}}(X_i, \mathbf{Pa}_i^{\mathcal{G}} \ : \ \mathcal{D})$

When assuming Dirichlet priors this leads to the same closed form formula derived in Eq. (2.11). The only difference being that the counts $M[\mathbf{U}]$ and $M[x_i^j, \mathbf{u}]$ are tallied only over the samples $X_i \notin \mathcal{M}[m]$.

$$\mathrm{FamScore}_{\mathrm{Int}}(X_i, \mathbf{Pa}_{X_i} \ : \ \mathcal{D}) = \log \prod_{\mathbf{u} \in Val(\mathbf{Pa}_{X_i})} \frac{\Gamma(\alpha_{x_i|u})}{\Gamma(\alpha_{x_i|u} + M[\mathbf{u}])} \prod_{x_i^j \in Val(X_i)} \frac{\Gamma(\alpha_{x_i^j|u} + M[x_i^j, \mathbf{u}])}{\Gamma(\alpha_{x_i^j|u})} \tag{4.6}$$

4.2.3 Inferring causality with mutational data

The score of Eq. (4.5) is not *structure equivalent*: The score of two equivalent graphs, G and G' is no longer guaranteed to be the same. This should not come as a surprise, the score was derived in an intent to use mutational data to differenciate between equivalent graphs. We use the equivalent graphs $X \rightarrow Y$ and $X \leftarrow Y$ to demonstrate this point

Example 4.2.1: Assume our domain contains the variables (X, Y) and we measure the following samples:

$$\begin{aligned}\mathcal{D} = \{&(0,0),(0,1),(1,1),(1,1),(1,0),(0,0),(do(X=1),1),\\ &(do(X=1),1),(do(X=1),0),(do(X=0),0)\}\end{aligned}$$

For this data, the score for the graph structure $X \rightarrow Y$ is -6.46. In contrast, the score for the graph structure $Y \rightarrow X$ is : -6.78. Since the score for $X \rightarrow Y$ is better, we conclude that it is more likely that X causes Y.

∎

While mutations help determine the causal direction of some edges, usually mutations only help orient part of the edges, leaving the causal direction of the others undetermined. This motivates developing a notation of equivalence suited for the mutational setting.

Definition 4.2.2: For a set of interventions $\mathcal{M}$, we say that the graphs $\mathcal{G}^1$ and $\mathcal{G}^2$ are $\mathcal{M}$-equivalent, if for any set of mutations $m \in \mathcal{M}$ (always including $m = \emptyset$) the graph structures $\mathcal{G}^1_{do(m)}$ and $\mathcal{G}^2_{do(m)}$ are equivalent. ∎

Using the empty set of mutations, $\mathcal{M}$-equivalence implies equivalence as defined in Definition 2.2.4. The notion of $\mathcal{M}$-equivalence is more restrictive refinement of Definition 2.2.4 in which a larger set of edges are compelled.

Definition 4.2.3: We say that an edge $X \rightarrow Y$ in $\mathcal{G}$ is $\mathcal{M}$-compelled for a set of interventions $\mathcal{M}$ if all graphs that are $\mathcal{M}$-equivalent to $\mathcal{G}$ contain the directed edge $X \rightarrow Y$.∎

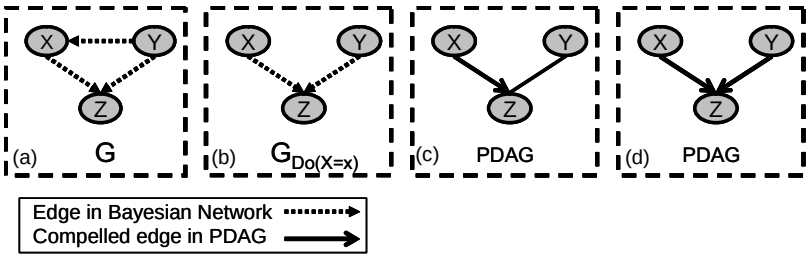

Figure 4.6: v–structures in $\mathcal{G}_{do(m)}$ are compelled: (a) $\mathcal{G}$ does not have a v-structure. (b) $Y \rightarrow X$ is removed in $\mathcal{G}_{do(X=x)}$ creating a v-structure. (c) The edge $X \rightarrow Z$ is adjacent to a mutated variable and thus compelled by Theorem 4.2.4. (d) The edge $Y \rightarrow Z$ is compelled by the second propagation rule in Figure 2.4

.

Theorem 4.2.4: The following edges are $\mathcal{M}$-compelled in $\mathcal{G}$.

1. All edges participating in a v-structure in $\mathcal{G}$.
2. For each set of mutations $m \in \mathcal{M}$, all edges entering or leaving any variable X mutated in m.
3. All edges compelled by the repeated application of the propagation rules specified by Figure 2.4.

Proof:

1. Using $m = \emptyset$, all graphs must be equivalent to $\mathcal{G}$ and therefore contain the same v-structures.
2. Assume X is mutated in m and the edge $X \rightarrow Y \in \mathcal{G}$. Then the skeleton of $\mathcal{G}_{do(m)}$ contains the edge $X - Y$. For any graph $\mathcal{G}'$ with the edge $X \leftarrow Y$, the skeleton of $\mathcal{G}'_{do(m)}$ does not contain the edge $X - Y$. Therefore $\mathcal{G}_{do(m)}$ and $\mathcal{G}'_{do(m)}$ are not equivalent. The reasoning for an edge $X \leftarrow Y \in \mathcal{G}$ is similar.
3. Using the same reasoning as explained in Section 2.2.2, if these edges would not be compelled, a new v-structure not in $\mathcal{G}$ or $\mathcal{G}_{do(m)}$ would form.

∎

Note that the set of rules specified in Theorem 4.2.4 also compels all v-structures in $\mathcal{G}_{do(m)}$. The only way a new v-structure can form in $\mathcal{G}_{do(m)}$ is if $\mathcal{G}$ contains the edges $X \rightarrow Z$, $Y \rightarrow Z$ and $Y \rightarrow X$ and in $\mathcal{G}_{do(m)}$ the edge $Y \rightarrow X$ is removed. This happens only if X is mutated in m, therefore $X \rightarrow Z$ is compelled based on the second rule of Theorem 4.2.4. This forms the structure $X \rightarrow Z - Y$, then the second propagation rule in Figure 2.4 compels the edge $Y \rightarrow Z$.

Corollary 4.2.5: *The $\mathcal{M}$-equivalence class of $\mathcal{G}$ can be represented using a PDAG, $\mathcal{P}$, in which all $\mathcal{M}$-compelled edges are directed.*

Theorem 4.2.6: For any $\mathcal{M}$-equivalent graphs $\mathcal{G}^1$ and $\mathcal{G}^2$ and any dataset D, s.t. $\mathcal{M}[D] \subseteq \mathcal{M}$, the following holds:

$$\text{score}_{\text{Int}}(\mathcal{G}^1 \ : \ D) \ = \ \text{score}_{\text{Int}}(\mathcal{G}^2 \ : \ D)$$

We note that our definition of $\mathcal{M}$-equivalence is sound but not complete. While all graphs in a $\mathcal{M}$-equivalence class are score equivalent, our definition of $\mathcal{M}$-equivalence does not necessarily include all score equivalent graphs.

We begin with a number of lemmas that aid the proof of Theorem 4.2.6.

Lemma 4.2.7: *Let $\mathcal{G}^1$ be a DAG containing the edge $X \rightarrow Y$ and $\mathcal{G}^2$ be a DAG identical to $\mathcal{G}^1$ except that $X \rightarrow Y$ is replaced with $X \leftarrow Y$. Then $\mathcal{G}^1$ and $\mathcal{G}^2$ are $\mathcal{M}$-equivalent iff $X \rightarrow Y$ is a covered edge in $\mathcal{G}^1$.*

This is variation of Theorem 2.2.8, adjusted to the concept of $\mathcal{M}$-equivalence.

Lemma 4.2.8: *Let $\mathcal{G}^1$ be a DAG containing the edge $X \rightarrow Y$ and $\mathcal{M}$ be a set of interventions s.t. neither X nor Y are mutated in $\mathcal{M}$. Let $\mathcal{G}^2$ be a DAG identical to $\mathcal{G}^1$ except that $X \rightarrow Y$ is replaced with $X \leftarrow Y$. If $X \rightarrow Y$ is a covered edge in $\mathcal{G}^1$ then for any $m \in \mathcal{M}$ the graphs $\mathcal{G}^1_{\text{do}(m)}$ and $\mathcal{G}^2_{\text{do}(m)}$ are equivalent.*

Proof: (of Lemma 4.2.8) Using Lemma 4.2.7 it is enough to show that $\mathcal{G}^1_{do(m)}$ and $\mathcal{G}^2_{do(m)}$ are identical except for the orientation of the edge between X, Y and that $X \rightarrow Y$ is a covered edge in $\mathcal{G}^1_{do(m)}$. The construction of $\mathcal{G}_{do(m)}$ involves removing from $\mathcal{G}^1$ all edges incoming into mutated variables. The parent sets of every variable $Z \in \mathcal{M}[m]$ are equal: $\mathbf{Pa}_Z^{\mathcal{G}^1} = \mathbf{Pa}_Z^{\mathcal{G}^2} \Rightarrow$ the exact same set of edges is removed in both graphs $\Rightarrow$ the graphs remain identical apart from $X - Y$. Since neither X nor Y are mutated, their parent sets remain untouched, therefore $\mathbf{Pa}_X^{\mathcal{G}^1} = \mathbf{Pa}_X^{\mathcal{G}^1_{do(m)}}$ and $\mathbf{Pa}_Y^{\mathcal{G}^2} = \mathbf{Pa}_Y^{\mathcal{G}^2_{do(m)}}$. Thus if $X \rightarrow Y$ is covered in $\mathcal{G}^1$, it is also covered in $\mathcal{G}^1_{do(m)}$. ∎

Lemma 4.2.9: *Let $\mathcal{G}^1$ and $\mathcal{G}^2$ be any pair of $\mathcal{M}$-equivalent graphs for a set of mutations $\mathcal{M}$. Then there exists a sequence of $|\Delta(\mathcal{G}^1, \mathcal{G}^2)|$ distinct edge reversals applied to $\mathcal{G}_1$ with the following properties:*

- *After each reversal, the resulting graph $\mathcal{G}$ is $\mathcal{M}$-equivalent to $\mathcal{G}^2$.*
- *After all reversals, the resulting graph is identical to $\mathcal{G}^2$.*
- *Each edge reversed in $\mathcal{G}$ is an covered edge.*

Proof: This lemma is simple consequence of Theorem 2.2.8 which provides us with a sequence of covered edges $X_1 \rightarrow Y_1, \ldots, X_k \rightarrow Y_k$ that define a sequence of single edge reversals between equivalent structures $\mathcal{G}^1, \ldots, \mathcal{G}^k = \mathcal{G}^2$. For $i = 1, \ldots, k$ we have $X_i \rightarrow Y_i \in \Delta(\mathcal{G}^1, \mathcal{G}^2) \Rightarrow$ therefore the edge is not $\mathcal{M}$-compelled $\Rightarrow$ neither X_i nor Y_i are mutated in $\mathcal{M}$ (since all edges incident to mutated variables are $\mathcal{M}$-compelled). Using Lemma 4.2.8 we conclude that for each $m \in \mathcal{M}$ the graphs $\mathcal{G}_{i-1,do(m)}$ and $\mathcal{G}_{i,do(m)}$ are equivalent, therefore by definition, the graphs $\mathcal{G}_{i-1}$ and $\mathcal{G}_i$ are $\mathcal{M}$-equivalent. ∎

We are finally ready to prove Theorem 4.2.6:

Proof: As a consequence of Lemma 4.2.9 it is enough to prove that the intervention score is equal for any two $\mathcal{M}$-equivalent graphs that differ only in the orientation of a single covered edge. Let $\mathcal{G}^1$ and $\mathcal{G}^2$ be two such graphs and $X - Y$ the edge that they differ on. Since all edges adjacent to mutated variables are compelled, neither X nor Y are in $\mathcal{M}$. In this case the proof follows identically to regular notation of equivalence, see Theorem 2.3.12. ∎

4.2.4 Empirical results

We chose the *S. Cerevisiae* Galactose dataset [53] to empirically compare between the BDe score and the Intervention score. This dataset was created to study the biochemical pathway of galactose utilization and its regulation. The dataset is well suited for testing the potential advantages of the intervention score since it includes perturbations of all the genes in a small and well studied pathway. These perturbations include deletions of both metabolic and regulatory genes, all whom participate in the Galactose pathway. Furthermore, this is the only public dataset with 8 samples measured for each mutant strain (most datasets contain 1-3 samples per strain).

Galactose utilization converts galactose (transported into the cell by GAL2) into glucose-6-phosphate though a series of enzymes (GAL1, GAL5, GAL7 and GAL10). The regulatory genes GAL3, GAL4, and GAL80 exert tight transcriptional control over the pathway genes, and to a certain extent each other. The relations in the pathway are depicted in Figure 4.7. GAL4 is a DNA-binding factor that strongly activates transcription of the GAL genes. In the absence of galactose, GAL80 binds GAL4 and inhibits its activity. When galactose is present in the cell, GAL3 binds to GAL80 and represses its activity, thus freeing an active form of GAL4.

The Galactose dataset [53] measures the expression profiles of wild-type and nine genetically altered yeast strains, each such strain containing a deletion of a single GAL gene. Each strain is measured in two different growth conditions: galactose rich and galactose absent, resulting in 20 different conditions related to the galactose pathway. Four samples were measured from each condition, resulting in a dataset with 80 samples. We constructed a dataset composed of the nine galactose genes discretized into two values (the dataset is available at URL). We added an extra variable corresponding to the medium type (galatose rich or depleted). We re-sampled the data using a 200-fold bootstrap and applied our Bayesian network learning algorithm twice to each sampled dataset, once using the BDe score and once using the intervention score.

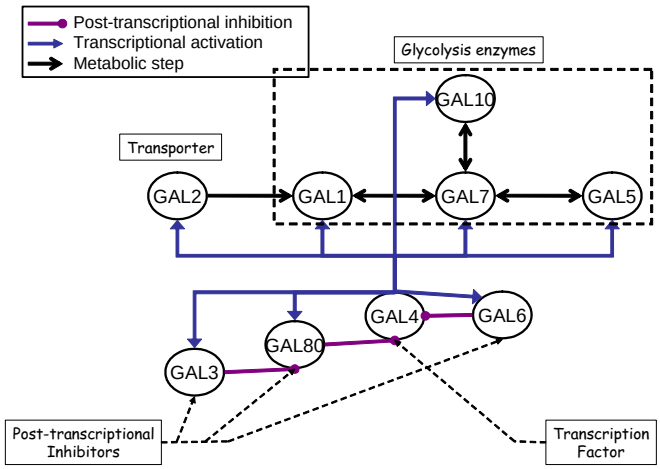

Figure 4.7: Galactose utilization pathway.

We compared gene mate features with confidence > 0.7 and edge features with confidence > 0.5 between the two scores. The intervention score has a considerably larger number of features in both categories: 20 verses 9 gene mates and 16 verses 4 edge features. Histograms comparing the distribution of feature confidences between the two scores appears in Figure 4.8. Since the equivalence class for the intervention score orients many more edges, it is not surprising the intervention score has 4 times as many edge features. We speculate that the preference in edge orientations leads to a narrowing of the set of paths searched and thus the larger number of gene mates. Furthermore, the graphs inferred using the intervention score are typically denser, with an average of 2.3 parents per variable verses an average of 1.6 parents per variable for the BDe score.

In our comparison we used a strict definition for "correct" gene mates: a pair of genes connected by an edge in Figure 4.7. We did not count relations with the medium variable as it was not included in our reference graph. The BDe score captured 4/7 correct gene mates: two transcriptional links (GAL4 – GAL10, GAL4 – GAL3) and two metabolic steps (GAL7 – GAL10, GAL7 – GAL5). The intervention score captures all the gene mates inferred by the BDe score (with at most 1% difference in the confidence level) and a few more giving it 8/15 correct gene mates. The gene mates (as well as edge features) for each score can be examined in Figure 4.9

The big advantage of the intervention score comes into play when comparing the edge relations. The BDe score captures only one correct feature GAL4 $\rightarrow$ GAL3 and three incorrect edges from GAL10 to (GAL1, GAL3 and GAL80). Using the intervention score, almost all correct gene mates are also correctly oriented edge features, with the exception of (GAL4 – GAL5) which is not oriented in any direction. More striking, contrary to the BDe score, the intervention score captures almost all of the transcriptional edges from GAL4 to its targets. There is a high confidence directed edge from GAL4 to each of GAL1,GAL2,GAL3,GAL7,GAL10,GAL80, and only a link to GAL6 is missing.

GAL3 and GAL80 both have a regulatory role and indeed we found a number of outgoing

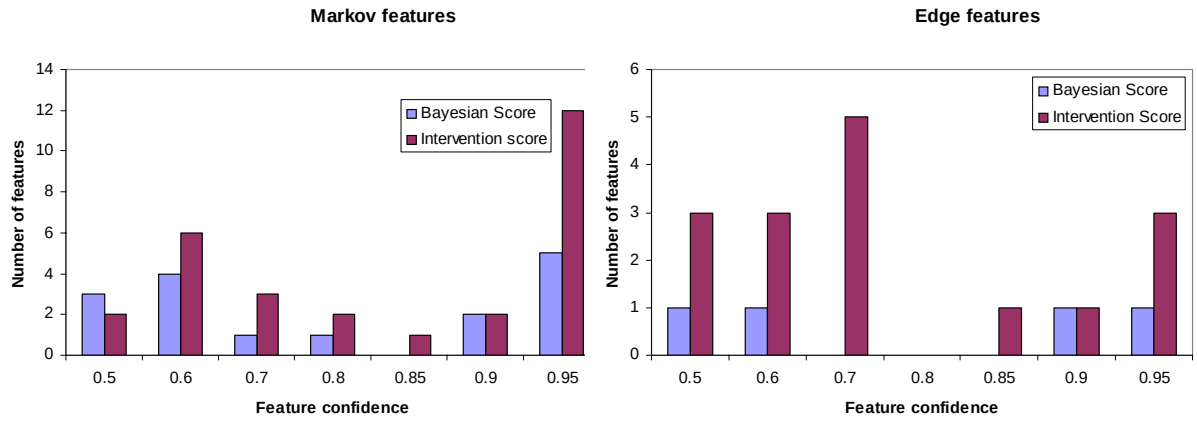

Figure 4.8: Comparison of feature distribution between the Bayesian and the intervention scores

edges from these regulators: (GAL80 → GAL10, GAL80 → GAL7, GAL80 → GAL3 and GAL3 → GAL2). While their regulatory effect is indirectly mediated through GAL4, this is mediation is executed in a post-transcriptional manner. The post-transcriptional protein-protein interactions between GAL4 – GAL80 and GAL80 – GAL3 are not observed in gene expression measurements. Therefore, the regulation of each GAL gene should be a logical function of the gene expression of GAL3,GAL4 and GAL80. We might expect the correct graph structure to contain 3 parents (GAL3, GAL4 and GAL80) for each GAL gene, but 80 samples are not enough to learn so many parameters. It is interesting to note that the strength of signal correlates well with the distance in the molecular chain of events, GAL3 → GAL80 → GAL4 → target. In our inferred network, GAL4 with 6 targets has the largest number of regulatees, next is GAL80 with 3 targets and GAL3 has one target.

Both scores consistently made the “mistake” of inferring that GAL10, an enzyme in the metabolic pathway, regulates other genes. Since the “mistake” is consistent, it is possible that this indeed correlates with a real biological signal. Looking back at the raw expression profiles of the GAL10 mutant strain, indeed in the presence of galactose, the expression of the genes inferred to be regulated by GAL10 (GAL1, GAL3, GAL7, GAL80) significantly changes. GAL10 is an enzyme responsible for processing Gal-1-P and the metabolite accumulates in its absence. The analysis done by Ideker et. al. [53] reached similar predictions for GAL10’s regulatory role. They hypothesized that the build up of Gal-1-P triggers some other regulatory mechanism which affects the expression of the GAL genes. They performed an experiment with a double mutant of GAL1 and GAL10 in which Gal-1-P does not build up. Consistent with their hypothesis, the expression of the GAL genes did not change in this double mutant. This is yet another demonstration that care must be taken when interpreting networks constructed from expression profiles. It is indeed true that GAL10 indirectly regulates the GAL genes, by causing Gal-1-P accumulation.

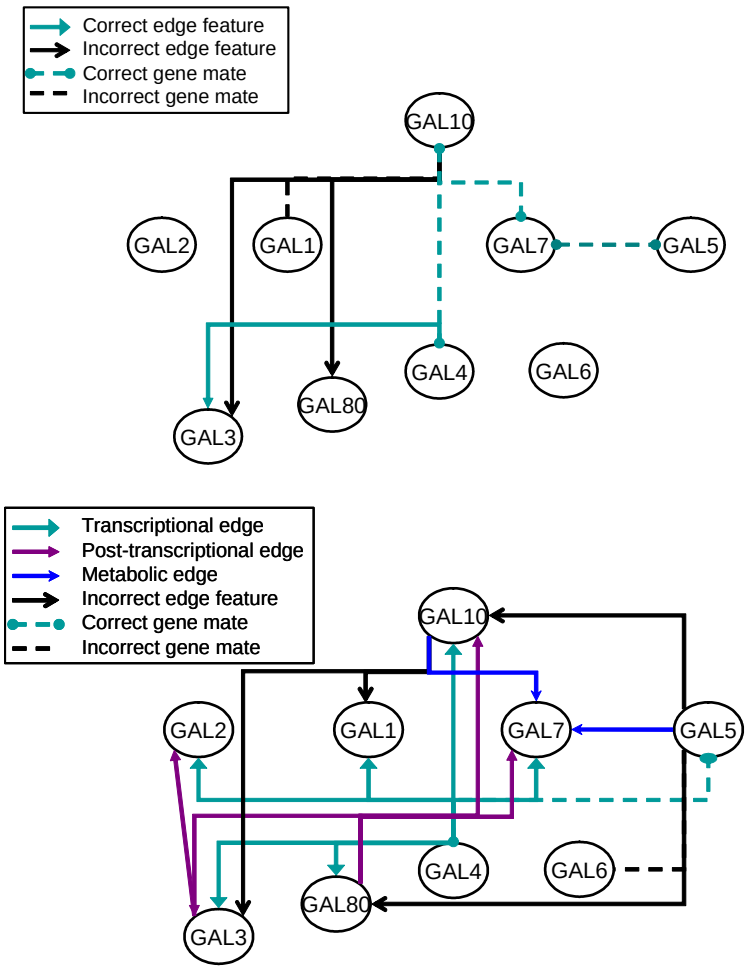

Figure 4.9: The gene mate and edge features inferred for the BDe score (top) and intervention score (bottom). The gene mates that do not have a corresponding edge relation are draw as a dashed line, all edge relations are also gene mates. The arrow type corresponds to the biological interpretation of the edge.

4.2.5 Discussion

While our proposed intervention score might be a first step in modeling genetic mutations within the framework of Bayesian networks, it suffers from a number of serious drawbacks. First, most gene expression datasets contain very few samples of each mutation. When the number of genes and samples is very large and the number of samples for each mutation is small (e.g. in the Compendium [51] dataset there only one sample per mutation for 276 mutations), ignoring the mutation causes only a minor change in the intervention score. This is contrary to the fact that a mutation is a very significant event. Our biological assumption is that the mutation is the *cause* for a large part of the differential expression observed in the mutated sample. This very strong assumption is currently not modeled into our intervention score. It is important to develop new methods and scores that give the mutation its key role in explaining the sample. A possible approach is to search for networks which "prefer" a causal path from the mutated gene to each of the significantly differentially expressed genes. Recent work by Yeang [102] suggests a model and algorithm along these lines.

The second drawback is that an ideal intervention is not a realistic model for a biological mutation. The cell is a very robust system which remains fit while withstanding many mutations. When one gene fails (e.g. gene mutated) the molecular network usually has back-up genes or back-up pathways that replace its role. This feature, called genetic buffering, is central to the proper functioning of the cell. One example of such a backup system is izozymes. These are formed when gene duplication creates two copies of a gene and evolotion differentiates their regulation, function or both. While these genes might have evolved into separate role, they remain close enough to substitute for each other. Another more concrete example (in *S. Cerevisiae*) is Dig1 and Dig2, both inhibitors of invasive growth by binding Ste12. Roberts et. al. [79] show that a double mutation is needed for a pronounced change in phenotype.

Mutations behave even less "ideally" in the case of over-expression. Membranes separate the cell into many different compartments. When a protein is significantly over-expressed, it often begins accumulating in cellular compartments from which it is absent under normal conditions. In these compartments it can now interact with proteins from which it is separated from under normal conditions. To our knowledge there is no network learning algorithm that takes genetic buffering into account.

In this section we presented the importance of modeling mutations into the network learning algorithms. We provided both theoretical motivation and empirically showed the advantage of taking the mutations into account. We used the Galactose dataset [53] to show that when using the standard BDe score most of the regulatory relations in the galactose pathway are missed. We demonstrated that even a simple adjustment to the score leads to the reconstruction of GAL4's regulatory role for almost all GAL genes from the same dataset. Our simplistic score and the strong assumptions associated with it are only a first step in properly modeling genetic mutations for the task of molecular pathway reconstruction.

Chapter 5

Focusing on Regulation - MinReg

In this chapter we demonstrate how we adapt the foundations of Bayesian networks to focus on the learning of regulatory relations between genes. In Section 5.1, we provide motivation and define a simplified network model for gene regulation: MinReg In Section 5.2, we describe the learning of such models from data. Section 5.3 provides technical details of our algorithm including theoretical guarantees on its performance. In Section 5.4, we present an automated methodology to interpret the resulting model and in Section 5.5, we demonstrate this on *Saccharomyces cerevisiae* gene expression data. Section 5.6 offers a more systematic evaluation of our the algorithm's performance.

5.1 A Regulation Graph

The common approach adapted by many computer scientists to the construction of gene networks is to attempt useing gene expression data to reconstruct a detailed graph in which the parents of each gene are the transcription factors that directly regulate it. In Chapter 3 we provided rationale why this goal is impractical: when learning a network model over a large number of variables, the number of possible solutions is prohibitively large. Not only is it computationally infeasible to search for the optimal model, even given the optimal solution, it is bound to overfit the data and contain many spurious artifacts. Furthermore, many transcription factors are activated post-transcriptionally, therefore their activity levels are not observed in gene expression data. Indeed, many researchers do not believe it is possible to reconstruct regulation from gene expression data alone.

In Chapter 3 we presented a Bayesian network methodolgy that reconstructs biological relations between genes and applied it to *Saccharomyces cerevisiae* gene expression data. Contrary to our expectations, only 4% (see Figure 3.4) of the inferred relations involved a transcription factor and its target gene. Are there so few regulatory relations because our specific model and learning algorithms are unsuited for the task, or, is there an inherent problem to capture regulatory events from gene expression data?

The space of possible Bayesian networks is so vast that even if regulatory signals exist in gene

expression data, detecting them is like finding a needle in haystack - the existing regulatory signal often gets lost among many spurious and non-regulatory relations. To develop an algorithm that focuses on regulatory relations, we invoke a number of biologically-motivated restrictions that simplify the model and significantly reduce the space of possible networks. A good model should be simple enough to be robustly inferred from the number of samples at hand, while capturing the essence of the true regulatory signal.

Our first restriction exploits prior biological knowledge to limit the possible parents in the network to a set of candidate regulators $\mathcal{C}$, i.e., $\mathbf{Pa}_X \subset \mathcal{C}$. The chosen candidate set consists of known and putative regulators in the organism being studied. Thus, our inference focuses on finding which candidate regulators are *actively* regulating other genes in the data, instead of finding which genes function as regulators. Unlike previous approaches [93], which limit themselves to transcription factors, we expand our set of candidates to proteins involved in different aspects of regulation, namely transcription factors, signal transducers and protein kinases. While this restriction may miss "unknown" regulators, it is justified because it enhances the possibility to find regulatory relations.

To justify our use of signal transducers and protein kinases, we stress that in order to capture a regulation event in gene expression data, we must observe changes in the expression of both the regulator and the *regulatee* (target gene). Unfortunately, while transcription factors directly control transcription, the factors themselves are frequently regulated post-transcriptionally, and their expression levels are often too low to allow reliable detection with microarrays. In such cases, we cannot observe the change in their regulatory activity in gene expression profiles. On the other hand, Signaling molecules, are expressed at significantly higher levels and may be regulated transcriptionally. Thus, we can capture regulatory relations indirectly, by the change in the expression levels of the signaling molecule and its indirect target regulatee genes.

The next restriction constrains the structural properties of the regulation graph. We seek a regulation model (Bayesian network) in which only a limited number of genes have an outdegree greater than zero. Given the set of candidate regulators, we wish to find a small sub-set of *active regulators* which are globally predictive of the gene expression observed in the data. This is based on both biological and statistical motivation:

1. Given gene expression dataset, only a small fraction of the genome is directly involved in regulating transcription, and each such "master regulatory gene" may affect the transcription of many genes.
2. Only when a gene consistently scores high as a parent for many genes, do we believe it indicates a true signal. An occasional high score as a parent of a single gene is attributed to spurious chance.

A *regulation graph* is a Bayesian network with the above restrictions on its structure. The *regulators* of gene X (denoted $\mathbf{Pa}_X$) are its parents in the graph. Thus, each edge connects a *regulator* to its *regulatee*. We now formally define our model of gene regulation:

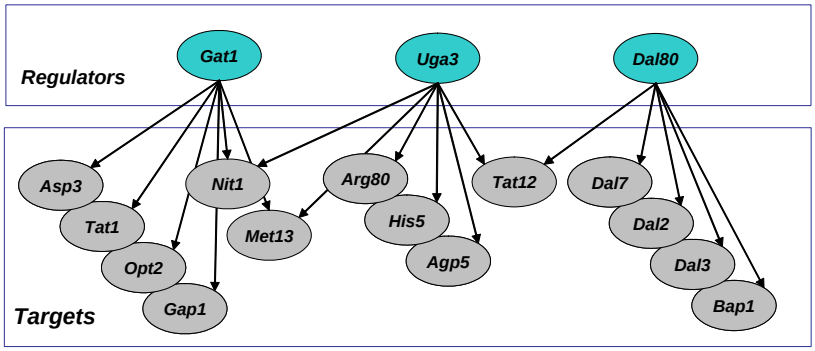

Figure 5.1: Two layer regulation graph. The top layer is associated with the regulators and the bottom layer is associated with regulatees. The key concept behind the regulation graph is a small number of regulators, each with many targets. Note, the illustrated GATA regulation was automatically inferred by our algorithm

Definition 5.1.1: Given a set of genes - $\mathcal{X}$, a set of *candidate regulators*- $\mathcal{C}$ and the constants d and k, we define a *regulation graph*, $\mathcal{G}$ to be a Bayesian network over $\mathcal{X}$ so that:

- All regulators (parents) belong to the candidate set: $\forall X, \mathbf{Pa}_X \subset \mathcal{C}$.
- The number of regulators for each gene (indegree) is bounded by d: $\forall X, |\mathbf{Pa}_X| \leq d$.
- The total number of regulators in the model is bounded by k: Denote $\mathcal{R}$ to be the set of all regulators in the network, then $|\mathcal{R}| \leq k$.

∎

The graph structure is best visualized as a two layer graph: in the top layer, a small set of regulators (chosen from a large set $\mathcal{C}$) and in the bottom layer, all other genes (see Figure 5.1). In our experiments, d ranges between 3 to 5, k ranges between 20 to 50, $|\mathcal{C}|$ is in the order of hundreds, and $|\mathcal{X}|$ is in the order of thousands

5.2 Learning

In this section we present *MinReg*, a greedy algorithm that reconstructs a *regulation graph* (Definition 5.1.1) from gene expression data. Recall, Bayesian networks (Definition 2.1.1) consist of two components, one structural and one mechanistic component: the expression level of each regulatee gene is a probabilistic function of its regulators. Our learning algorithm is based on this mechanistic component: given a *training set* $D = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$ of independent instances of $\mathcal{X}$, our goal is to find a regulation graph that best explains the data. A good model is one in which the regulators predict the expression level of their targets.

5.2.1 Optimization Problem

Since a regulation graph is a Bayesian network, the straightforward approach to learning the graph structure would be to use the heuristic greedy hillcimbing search (see Section 2.3.2) typically used for this task. This involves traversing the space of legal models in a greedy fashion using local operators such as adding, removing or reversing a single edge. At each step, the operation that best improves the score is chosen.

Unfortunately, this approach could fail, because the limited number of regulators specified by the regulation graph would be quickly used up. Consider the following scenario: We wish to search for a regulation graph over 2000 genes, limited to 30 regulators. We begin with the empty graph and in a greedy fashion add the optimal edge at each iteration. In many of these iterations, a new regulator is added to the regulator set $\mathcal{R}$. Therefore, after little over 30 iterations, no new regulators could be added to $\mathcal{R}$ and all subsequent legal steps would only involve adding edges from regulators already in $\mathcal{R}$. While these regulators might be the best parents for a small set of genes, thousands of other genes remain unexplained in the model.

Contrary to learning regular Bayesian networks, the choice of regulators for a gene X is no longer independent of the regulators chosen for other genes. Since the total number of regulators in the model is limited to k, choosing a regulator for one gene can limit the choice of regulators for other genes. A regulator should be added to $\mathcal{R}$ only when it is a good choice for many genes concurrently.

To our advantage is that when the regulator set $\mathcal{R}$ is given, finding the optimal regulation graph constrained to $\mathcal{R}$ is polynomial, and for most practical cases efficient. First, for any given variable, there is a small number of possible parent sets: There are only $\binom{k}{d}$ options for $\mathbf{Pa}_X$: when d ranges between 3 to 5 and k ranges between 20 to 50, it is quick to calculate the local score for all possible parents sets and choose the highest scoring set.

$$\mathbf{Pa}_X = \text{argmax}_{P \subset \mathcal{R}, |P| \leq d} \textit{Score}(X; P) \tag{5.1}$$

While $\mathcal{R}$ is not given, the MinReg algorithm will use this property to efficiently search of a good set of regulators. In comparison, there are $\binom{n}{d}$ possible parent sets for Bayesian networks bounded by an indegree of d. Typically $n >> k$, thus optimizing the local score when there is no candidate set is significantly more expensive.

Definition 5.2.1: We define the *utility*, $F(\mathbf{R})$ of a regulator set $\mathbf{R}$ as:

$$F(\mathbf{R}) = \sum_{X \in \mathcal{X}} \max_{P \subset \mathbf{R}, |P| \leq d} \textit{Score}(X; P) \tag{5.2}$$

∎

Calculating the utility of a regulator set $\mathbf{R}$ is quick and it closely approximates (upper bound) the

score of the optimal network constrained to $\mathbf{R}$. It scores a graph structure resulting from independently choosing the optimal parents for each variable, but this structure may contain cycles. A high scoring Bayesian network can be constructed by transforming this structure into an a-cyclic graph with minimal effect to the score. Cycles can form only between the genes in $\mathcal{R}$, therefore, acyclicity can be resolved within a subgraph involving only k nodes. The complexity of solving this problem is constant in n, though exponential in k. In most practical cases, only a few short cycles form and the optimal solution can be easily found. In comparison, cycles can form among any n variables in a Bayesian network, thus, resolving cyclicity can be exponential in n. While 2^k is feasible, 2^n is not ($k << n$).

In our typical setting where k ranges between 20 to 50 and $n \geq 2000$, the difference in score after breaking the cycles is negligible. With high probability this difference would not change our choice of regulating set $\mathcal{R}$. Therefore, for the remainder of this section we ignore the issue of cyclicity. We treat $F(\mathbf{R})$ as a scoring function that measures the quality of regulator sets. This implies a new optimization problem to find a small set of regulators, $\mathcal{R}$, which maximize this score:

Definition 5.2.2: The *Best Regulator Set* problem: Given a set of genes $\mathcal{X}$, a dataset D, a set of candidate regulators $\mathcal{C}$, and the constants d and k, we wish to find

$$\mathcal{R} = \text{argmax}_{\mathbf{R}|\mathbf{R}\subset\mathcal{C},|\mathbf{R}|\leq k} F(\mathbf{R}) \tag{5.3}$$

∎

This problem is conceptually similar to the *Set Cover* problem, a classical hard problem. The challenge is that the regulator set $\mathcal{R}$ needs to chosen from a much larger candidate set $\mathcal{C}$ and there are $\binom{|\mathcal{C}|}{k}$ possible regulator sets. While there does not seem to be any efficient algorithm to find an optimal solution, in Section 5.2.2 we present an efficient algorithm that finds a model whose score is close to optimal.

What makes the problem hard is also what makes our solution robust. Consider the following thought experiment: A regulator r regulates genes $X_1, \ldots, X_{50}$. A typical training set consists of only a small number of noisy gene expression profiles. Therefore, due to spurious signal, $Y_i \neq r$ might be the highest scoring regulator for $X_i, i = 1 \ldots 50$. While r's score is inferior to Y_i, it is often not far behind. The basic principle behind our approach is that instead of choosing the top scoring candidate for each single gene, we choose a candidate that scores high for many genes. We believe a high score with many genes most likely originates from real biological signal in the data.

Scoring Function

Denote by $\mathbf{R}_\mathcal{G}$ the set of regulators in the structure $\mathcal{G}$. Recall, the score of an entire regulation graph $\mathcal{G}$, decomposes into sum over the local scores for each variable.

$$\text{score}(\mathcal{G} \ : \ D) = F(\mathbf{R}_\mathcal{G}) = \sum_i \textit{Score}(X_i; \mathbf{Pa}_{X_i}) \tag{5.4}$$

In previous chapters we used the BDe score (see Section 2.3.2) to score a set of regulators (parents) for a given gene. A key benefit of the BDE score is that it strongly penalizes the number of parameters in a model, thus, limiting the complexity of the optimal model. In Section 4.1.2 we argued that the BDe score is simply a cautious approximation of mutual information, that also penalizes the number of parameters.

The complexity of a regulation graph is inherently limited by its definition: Each gene is restricted to at most d regulators, limiting the number of parameters in the model accordingly. Therefore, we simply use *mutual information* for our scoring function.

$$\mathit{Score}(X; \mathbf{Pa}_X) = \sum_{x,\mathbf{u}} P(x, \mathbf{u}) \log \frac{P(x, \mathbf{u})}{P(x)P(\mathbf{u})} \tag{5.5}$$

By maximizing the mutual information between regulators and regulatees, we are minimizing the conditional entropy of the regulatees. Thus, given the values of the regulators, we minimize our uncertainty when predicting the values for the rest of the genes.

5.2.2 MinReg Algorithm

We now propose a greedy algorithm that searches for the best regulator set and its corresponding regulation graph: Begin with an empty set of regulators and an empty graph structure. At each iteration, for each possible candidate, we construct an increment regulator set by adding that candidate to the current regulator set. We calculate the score for each of the increment regulator sets and choose the one that gives the largest gain. Each time $\mathcal{R}$ is updated, we calculate the optimal regulation graph restricted to the current regulator set $\mathcal{R}$. We continue to iterate until no candidate contributes a significant gain to the score. A sketch of the MinReg algorithm is presented in Figure 5.2.

A crucial point is to correctly define the gain of a given regulator at each iteration. Note, we calculate mutual information between a variable and its regulating *set*. When considering a new candidate regulator c as a parent for a regulatee gene X, we measure not how much information c holds on X, but how much additional information c holds. Thus, each of the regulating parents provide a distinct contribution to the score.

Definition 5.2.3: We define the *marginal utility* of adding a regulator set $\mathbf{C}$ to an already chosen regulator set $\mathbf{R}$ as

$$F(\mathbf{C}|\mathbf{R}) = F(\mathbf{C} \cup \mathbf{R}) - F(\mathbf{R}) \tag{5.6}$$

∎

Using this terminology, at each iteration, we add the candidate regulator with the largest marginal utility. In order to sharpen the distinction between utility and marginal utility we provide Example 5.2.4 below.

Example 5.2.4: We wish to choose two optimal regulators for X between three candidates, Y_1, Y_2, Y_3

(In realistic cases we would have many more candidates). The table below lists their observed discetized expression over 16 samples.

Array #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Gene X	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	0
Reg Y_1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
Reg Y_2	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Reg Y_3	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	-1	0

The pairwise mutual information between X and each of the candidate regulators is: *Score*$(X;Y_1) = 0.439$, *Score*$(X;Y_2) = 0.358$, *Score*$(X;Y_3) = 0.157$. A greedy algorithm would first choose Y_1, the highest scoring regulator. A simple minded approach would then choose Y_2, since its mutual information is second best. Our algorithm works differently. It calculates the mutual information between X and Y_1, Y_2 together (same for Y_1, Y_3). The resulting calculations give *Score*$(X;Y_1,Y_2) = 0.532$ and *Score*$(X;Y_1,Y_3) = 0.778$. Our algorithm chooses the regulator that holds most additional utility, in our case Y_3. While Y_2 had higher mutual information, this information is overlapping with Y_1, both genes explain almost the same set of microarrays. On the other hand, Y_3 provides information on a different set of microarrays. ∎

5.3 Technical Details

5.3.1 Performance Guarantee

MinReg is a greedy algorithm. In each iteration, it chooses the best single regulator according to some local criteria and adds it to the model. But, this greedy approach does not necessarily lead to a global optimum. Can we characterize the situations in which the MinReg algorithm is lead astray? Consider the case where *Score*$(X;A)$ + *Score*$(X;B)$ is significantly less than *Score*$(X;A \cup B)$. In this situation, neither A nor B would be attractive enough to get selected by themselves in any of the greedy steps, where as their joint contribution may be significantly higher than any other combination of regulators. Thus, the greedy algorithm is mislead to choose an inferior regulator set.

In this section we argue that this characterizes the only situation in which our algorithm fails. We show that if we can bound the severity of such effects, we can derive a worse case error bound on the algorithm's performance: in this case, MinReg is an *approximation algorithm* guaranteed to find a solution which is not too far from the optimal. To formally prove this guarantee, we introduce the notation of *a-modular* functions.

Definition 5.3.1: [63] A function f is α-modular ($\alpha \geq 1$) if and only if $\forall Z, \mathbf{A}, \mathbf{R}$ the following holds:

$$f(\mathbf{A} \cup Z|\mathbf{R}) \leq f(\mathbf{A}|\mathbf{R}) + \alpha f(Z|\mathbf{R})$$

∎

Note that this is a generalization of sub-modular functions, (f is sub-modular for $\alpha = 1$). One might consider α as some measure on the convexity of f over the space of subsets from $\mathcal{X}$. For larger α, more "synergy" can be gained by joining sets together. We will show that if we can bound the amount of "synergy" between regulators, we can bound the error of our greedy algorithm accordingly.

Lemma 5.3.2: *The following are equivalent definitions of α-modular functions: ([63])*

1. *For any $\mathbf{S} \subseteq \mathbf{T}$ and $X \notin \mathbf{T}$ we have $f(X|\mathbf{S}) \geq \alpha f(X|\mathbf{T})$.*
2. *For any $\mathbf{S} \subseteq \mathbf{T}$ we have $f(\mathbf{V}|\mathbf{S}) \geq \alpha f(\mathbf{V}|\mathbf{T})$.*
3. $f(\mathbf{A}) + \alpha f(\mathbf{B}) \geq f(\mathbf{A} \cup \mathbf{B}) + \alpha f(\mathbf{A} \cap \mathbf{B})$

These equivalent formulations offer us another perspective: the marginal utilities of α-modular functions are "almost" (up to a factor of α) monotone decreasing. This fits our intuition that as $\mathcal{R}$ grows larger, the utility of adding new regulators diminishes.

Theorem 5.3.3: If F is an α-modular and monotone increasing function[1], then the MinReg algorithm (presented in Figure 5.2) is a polynomial time approximation algorithm for the Best Regulator Set: Denote by OPT, the score of the optimal solution and by GRD, the score of the solution found by the MinReg algorithm, then

$$(\alpha + 1)\mathrm{GRD} \geq \mathrm{OPT} \tag{5.7}$$

Proof: Our proof is by induction. For $k = 1$, the optimal solution is the best single regulator and therefore $\mathrm{GRD} = \mathrm{OPT}$. We assume that $(\alpha + 1)\mathrm{GRD} \geq \mathrm{OPT}$ for $k - 1$ and prove it for k.

Denote by $\mathbf{S}$ the regulator set of size k that achieves the optimal solution for F. Set $J = \mathrm{argmax}_{I \in \mathcal{C}} F(I)$, the best single regulator in $\mathcal{C}$ and $J' = \mathrm{argmax}_{I \in \mathbf{S}} F(I)$, the best single regulator in $\mathbf{S}$. Note, J is the first regulator chosen by the greedy algorithm.

We define the following sub-problem imitating the behavior of the greedy algorithm. Let $F'(\mathbf{Y}) = F(\mathbf{Y} \cup \{J\}) - F(J)$, our goal is to find a set of $k - 1$ regulators that optimize F' on $\mathcal{C} \setminus \{J\}$. This is exactly what MinReg does after it chooses J in the first iteration. We denote by OPT' and GRD' the scores for optimal and greedy solutions respectively to this new sub problem. It is easy to see that F' is a α-modular function and that our induction holds for F' as well.

By the inductive hypothesis, it suffices to show that the increment is α-modular, i.e.:

$$\mathrm{OPT} - \mathrm{OPT}' \leq (\alpha + 1)(\mathrm{GRD} - \mathrm{GRD}') = (\alpha + 1)F(J) \tag{5.8}$$

Since OPT' is at least as good as any solution of size $k - 1$, by definition:

$$\mathrm{OPT}' \geq F'(S \setminus \{J'\}) = F(S \setminus \{J'\} \cup \{J\}) - F(J) \tag{5.9}$$

[1]While the marginal utilities should be almost monotone decreasing, we want the function itself to be monotone increasing

By α-modularity:

$$\mathrm{OPT} = F(S) \leq F(S \setminus \{J'\}) + \alpha F(J') \tag{5.10}$$

Subtracting these two bounds give:

$$\mathrm{OPT} - \mathrm{OPT}' \leq [F(S \setminus \{J'\}) - F(S \setminus \{J'\} \cup \{J\})] + [\alpha F(J') + F(J)] \tag{5.11}$$

Monoticity of F implies that $F(S \setminus \{J'\}) \leq F(S \setminus \{J'\} \cup \{J\})$, therefore, the first bracket gives a negative contribution. Maximality of J implies that $F(J) \geq F(J')$, therefore the second bracket is $\leq (\alpha + 1)F(J)$. ∎

It remains to show that F is α-modular. Recall, F is a sum of local scoring functions that measure the mutual information between a gene and its regulators. If mutual information was an α-modular function, than F as a sum of α-modular functions would be α-modular function too. While mutual information is always monotone, it is not necessarily α-modular. Specifically, since synergy (the opposite of α-modularity) is known to play an important role in biological regulation, we do not expect mutual information to be α-modular in the gene expression domain. Fortunately, while regulators are synergistic for specific targets, F is a sum over thousands genes. Even if the synergy between two regulators is very high, this synergy would need to hold for many targets, otherwise it would average out when summing over all of $\mathcal{X}$. We empirically tested the synergy between regulators and groups of regulators for our dataset. In practice, by joining regulators together, the worse factor we encountered was 1.2, thus, we safely assume α-modularity of F with $\alpha = 2$ in our dataset.

5.3.2 MinReg Implementation

A general overview of the MinReg algorithm was presented in Section 5.2.2. Several details in MinReg's implementation lead to substantial speed-up of the naïve algorithm. Our implementation can generate a model over thousands of genes within minutes.

First, we define a function f_X for each gene X, $f_X(\mathbf{R}) = \max_{P \subset \mathbf{R}, |P| \leq d} \textit{Score}(X; P)$, this is the optimal contribution of X to F, restricted to a regulator set $\mathcal{R}$. Thus, we have 3 levels of scoring functions: *Score*, for a particular pair of gene and its regulators, f_X, the optimal score of a single gene, and $F(\mathbf{R}) = \sum_{X \in \mathcal{X}} f_X(\mathbf{R})$.

The naïve greedy algorithm has k iterations. In each iteration, $F(c|\mathcal{R})$ is calculated for all $c \in \mathcal{C}$. Each calculation of $F(c|\mathcal{R})$ requires calculating $f_X(c|\mathcal{R})$ for all $X \in \mathcal{X}$, thus, f_X is calculated $k|\mathcal{C}||\mathcal{X}|$ times. Calculation of f_X requires calculating *Score* for each of the $\binom{k}{d}$ possible sets of parents, while this is constant in k and d, in practice k could be large. We devise a number of heuristics based on α-modularity to reduce the number of times we need to calculate each of the 3 functions F, f_X, and *Score*.

We employ a branch and bound approach to $F(c|\mathcal{R})$: the idea is to use the α-modularity of F to filter out candidates with little potential. In the first iteration, for all $c \in \mathcal{C}$ we calculate

MinReg Algorithm
set $R = \emptyset$, $F = 0$
do ($i = 1 \ldots$) **set** $F' = F$ {
  //For each iteration find $c^* = \text{argmax}_{c \in \mathcal{C}} F(c \mid R)$
  foreach $c \in \mathcal{C}$ { **set** $R' = R \cup c$, $F'' = 0$
    foreach $X \in \mathcal{X}$ { **set** $\mathbf{P}_X = \emptyset$
      //greedily approximate $\max_{P \subset R', |P| \leq d}$ *Score*$(X; P)$
      for $j = 1 \ldots d$ {
        $\mathbf{P}_X = \mathbf{P}_X \cup \text{argmax}_{p' \in R' \setminus \mathbf{P}_X}$ *Score*$(X; \mathbf{P}_X \cup p')$ }
      $F''+ =$ *Score*$(X; \mathbf{P}_X)$ }
    if $F'' > F'$ **set** $c^* = c$, $F' = F''$ }
  $R = R \cup c^*$, $F = F'$ }
until $\forall c F(c \mid R) <$ threshold

Figure 5.2: Overview of the MinReg algorithm. The algorithm consists of two nested greedy loops. The external loop finds the optimal set R of k regulators. For each $X \in \mathcal{X}$ an internal loop finds an optimal set of parents $\mathbf{Pa}_X$.

Util$(c) = F(c) = \sum_{X \in \mathcal{X}}$ *Score*$(X; c)$. We organize the candidate regulators in a heap sorted by Util(c). At any given time, Util$(c) = F(c|\mathbf{A})$, for some $\mathbf{A} \subseteq \mathcal{R}$. The α-modularity of F ensures that αUtil$(c) \geq F(c|\mathcal{R})$ (see Lemma 5.3.2). In most cases, we expect the regulator with the highest marginal utility to be towards the top of the heap.

Once a new regulator is added to $\mathcal{R}$, the marginal utilities change and need to be recalculated. In each subsequent iteration, we traverse down the heap and only re-evaluate candidates for whom $\alpha *$ Util(c) is greater than the best marginal valuation found thus far, denoted $\hat{c}^*$. Each time $F(c|\mathcal{R})$ is calculated, we use this value to update Util(c) in the heap. Once we reach a candidate so that $\alpha *$Util$(c) < F(c^*|\mathcal{R})$ we stop traversing the heap, α-modularity ensures that none of the candidates beyond this point can be better than $\hat{c}^*$. While this branch and bound does not change the worst case complexity, for most practical cases only the few topmost candidates are examined in each iteration.

While the previous speed-up came at no loss in the quality of the final solution, the next two heuristics reduce accuracy. These heuristics are based on the assumption that *Score* is almost α-modular (though probably by a larger factor than F). While regulation is sometimes synergistic, functions such as XOR are rare in biology. We expect that even when synergy exists, it is bounded by a reasonable constant. More importantly, we expect the synergistic pairs are themselves uncommon. We assume the probability that two small randomly chosen sets of regulators are highly synergistic in relation to a random target is very small. Thus, as a close approximation, we treat *Score* as α-modular.

Similarly to how Util(c) approximates $F(c|\mathcal{R})$, we cache Util$_X(c)$ as an approximation of $f_X(c|\mathcal{R})$. Whenever $F(c|\mathcal{R})$ is calculated, we do so only approximately: $F(c|\mathcal{R}) = \sum_{X \in \mathcal{X}}$ Util$_X(c)$. In the first iteration we initialize Util$_X(c) = f_X(c)$, for each $c \in \mathcal{C}$ and $X \in \mathcal{X}$. In subsequent

iterations, we only recalculate $f_X(c|\mathcal{R})$ (and update $\text{Util}_X(c)$) for those X's whose parent set $\mathbf{Pa}_X$ changed in the previous iteration. This is especially effective in later iterations where $\mathbf{Pa}_X$ rarely changes.

Finally, instead of calculating f_X exactly, we approximate it using a greedy algorithm similar to Figure 5.2. We start with no parents and at each iteration add best parent, $argmax_{c \in \mathcal{R}} f_X(c|\mathbf{Pa}_X)$ to $\mathbf{Pa}_X$. This only requires $d|\mathcal{R}|$ calculations of *Score* instead of $\binom{|\mathcal{R}|}{d}$. Assuming α-modularity of *Score*, by applying the same proof used for Theorem 5.3.3, this greedy algorithm is a $\alpha + 1$ approximation of f_X.

5.4 Annotating Regulators

Learning a high scoring model is only the first step when analyzing data. Once a model is inferred, an interpretation that associates the abstract model with biological meaning is no less important. In this section we describe an approach to extract biologically meaningful features from the resulting regulation graph. We employ *local* features (e.g. Gene 1 regulates Gene 2) to deduce *global* ones (e.g. Gene 2 regulates cell wall organization). By using multiple local relationships, we devise a method to answer questions on the global role of regulators, such as "What is the biological process or molecular function that a gene regulates?".

While the basic building block of our model is the local *regulated* relationship, the main power of our framework lies in its global interpretation. Individual local relations must be treated with caution: they may only indicate spurious artifacts or co-regulation. As an alternative, we gain robustness by annotating each regulator according to prominent features of its *entire regulatee set*. For instance, if Gcn4, a activator of amino acid (AA) metabolism, is a regulator in our model. If the reconstruction is good, we expect Gcn4 to be the regulator of many of its AA metabolism targets. Therefore, the regulatee set of Gcn4's targets should be enriched with genes involved in AA metabolism, relative to a random selection of genes from $\mathcal{X}$. Reversing this logic, had Gcn4's role been unknown and we observe a significant enrichment of AA metabolism genes among its regulatees, we can deduce that *Gcn4 regulates AA metabolism*.

More formally, we denote by $\mathcal{X}_r$ the set of *regulatees* of a regulator r. We map r with specific biological processes (e.g. cell wall organization) and molecular functions (e.g. amino acid transporter) that are over represented in $\mathcal{X}_r$. For a regulator r and an annotation A, we calculate the fraction of genes in $\mathcal{X}_r$ associated with A and use the hypergeometric distribution to calculate a p-value for this fraction. In a nutshell, assume k out of n regulatees are associated with A. The hypergeometric p-value measures the probability of randomly drawing k genes annotated with A based on their fraction in the entire gene set $\mathcal{X}$. If there is a significant bias towards a certain process or function, we conjecture that r *regulates* that process. This process of *annotating regulators* is illustrated in Figure 5.3.

We devise a fully automated methodology based on the *Gene Ontology(GO)*. The Gene Ontology [16] provides a controlled vocabulary for the annotation of molecular function, biological

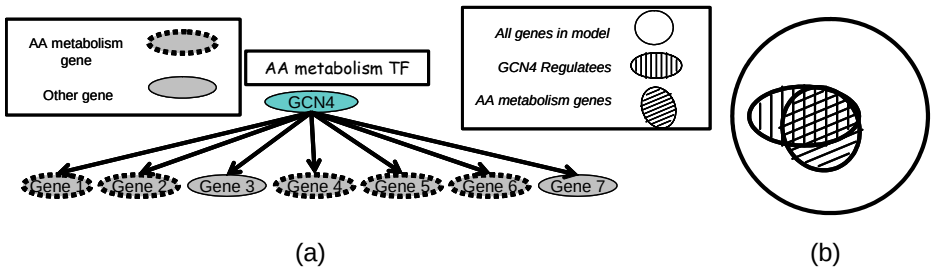

Figure 5.3: Annotating regulators. (a) We expect GCN4's targets to be enriched with AA metabolism genes. (b) Graphic representation of hypergeometric p-value. If the intersection between the set of all GCN4 regulatees and all genes annotated with AA metabolism is surprisingly large, GCN4 is annotated as regulating AA metabolism.

process and cellular location. These ontologies are embedded in hierarchy that permits queries at different levels of granularity: For example, one can find all the gene products that are involved in signal transduction, or one can zoom in on all the receptor tyrosine kinases. The GO terms are carefully defined, and the relationships between the terms are also detailed. An important quality of this structured ontology is that any gene annotated as a receptor tyrosine kinase will automatically be annotated with signal transduction as well. Furthermore, the *Saccharomyces Genome Database (SGD)* [11] has associated most characterized S.cerevisiae genes with one or more GO terms.

For a set of interest we systematically traverse all the nodes in the GO hierarchy down to a predefined depth, assigning a pvalue to each GO term. We perform a Bonferroni correction for multiple independent hypotheses and associate a regulator with its significant annotations. This functional annotation determines both the biological process regulated by the gene and the molecular mechanisms that are employed in the regulated process.

The automated method and calculated p-value provide an important improvement over the manual, time-consuming, anecdote-prone approach that was previously employed for this type of analysis. Note, in current databases many known annotations are missing and many genes are not annotated at all. Had these annotations existed in the database, not only would the p-values for existing associations be considerably higher, but also additional associations would be revealed. As our knowledge base grows and the quality of databases improve, we expect this automated approach will become even more effective.

Additional attributes can be tested for enrichment in the regulatee set. For example, we use motif detection methods to associate each regulator with a putative regulatory motif which occurs abundantly in the regulatory sequences upstream of the genes in $\mathcal{X}$.

Furthermore, we identify the logic (activation or inhibition) of regulation. Recall, the regulation model also has a mechanistic element which assigns each gene with a probabilistic function. Utilizing this function, we annotate each individual relation with one of 3 logic values: activation, inhibition or unknown. We assume a regulator exerts a coordinated effect on the subset of regulatees involved in the same process. Therefore, rather than characterizing individual regulatory relations,

we focus on the regulation of an entire process or response. As before, we use the hypergeometric distribution to look for regulatees of the same GO class enriched with of a particular logic.

5.5 Biological Results

We evaluated our algorithm on a dataset containing 358 samples combined from the Compendium [51] and stress [40] datasets. These datasets measure *S. cerevisiae* expression profiles under a wide variety of cellular conditions[2].

We automatically filtered non-informative genes and kept a subset of 3755 genes which displayed a significant change in gene expression in at least 15 samples. The data was normalized and discretized to 3 values: *down-regulated*, *no change* and *up-regulated*. We compiled a set $\mathcal{C}$ of 466 candidate regulators whose SGD [11] or YPD [21] annotations suggest a potential regulatory role in the broad sense: transcription factors, signal transducers and protein kinases that may have transcriptional impact. We also included genes described to be similar to such regulators. We excluded global regulators whose regulation is not specific to a small set of genes or processes. (see http://robotics.stanford.edu/ erans/module_nets/candidate_ regulators.html for full list).

We applied the MinReg algorithm to this dataset and continued to add regulators until no candidate significantly improved the score. Our current implementation of the MinReg algorithm required 19 minutes on an Intel III 1GHz processor. MinReg chose a set $\mathcal{R}$ of 45 regulators out of 466 candidates

We applied the regulator annotation procedure described in Section 5.4 and sorted the regulators according the pvalue of their most signficant association. We compared our derived annotations to those reported in the literature, indeed, our results corresponded well to previous findings, especially for significant pvalues. The associations for 8 of the top 10 regulators provided "proof of principle" (success or partial success, Table 5.5) by coinciding with well-known functional roles of these genes. Of the remaining two active regulators, we were able to assign a putative role to one previously uncharacterized gene, but failed to identify the correct role of the other[3]. Furthermore, we used the significant GO terms in order to focus our attention on individual regulatees which are more likely to represent true signal. Numerous individual regulator-regulatee relations of interest were examined, which lent support to our global analysis. Below we provide a number of detailed examples that demonstrate some of the abilities of our methodology.

Our functional assignment can be highly discriminative and comprehensive, providing us with an elaborate characterization of the regulator gene based on its regulatee set. For example consider Slt2, the MAP kinase that activates the cell wall integrity pathway. Our GO term derived annotation correctly predicts the biological process which Slt2 regulates (cell wall organization and biogenesis),

[2] [51] contains 276 deletion mutants from various functional classes and [40] contains 82 samples of responses to 12 different stress conditions.

[3] The results for the top 20 regulators are of similar quality: the associations for 13 regulators are proof of principle, 4 regulators were uncharacterized (see table 5.2) and the associations for 3 regulators are unsupported

Regulator (# regulatees)	Concise Annotation (YPD)	SGD (S) and YPD (Y) GO annotation biol process	MinReg's derived GO terms (score, #genes)	Verdict
Sst2 (99) signaling	Negative regulator of the mating pheromone signaling pathway by binding to Gpa1p and desensitizing it to pheromone	Adaptation to mating signal (S,Y) Signal transduction (S)	Mating (0,17) Signal transducer ($8.9e^{-05}$,7)	**Success**
Met28 (254) TF	Transcriptional activator of sulfur amino acid metabolism	Sulfur amino acid biosynthesis (S) Transcription regulation from Pol II promotor (S,Y)	Amino acid metabolism (0,24) Sulfur utilization ($1.2e^{-05}$,6)	**Success**
Gat1 (125) TF	GATA zinc finger transcription factor that activates genes needed to use non-preferred nitrogen sources	Amino-acid metabolism (Y) Other metabolism (Y) Pol II transcription (Y) Cell Stress (Y)	Protein biosynthesis (0,30) Amino acid metabolism (0.000106,10) Hydrolase, acting on carbon-nitrogen (but not peptide) bonds in linear amides (0.005208,3) Nitrogen starvation response (0.008642,2)	**Success**
Tea1(141) TF	Ty1 enhancer activator of the Gal4p-type family of DNA-binding proteins	Pol II transcription (Y)	Amino acid metabolism (0,15) Nitrogen metabolism ($7.9e^{-05}$,5) Urea cycle intermediate metabolism (0.000679,4)	**Novel function**
Uga3(73) TF	Transcriptional activator for 4-aminobutyric acid (GABA) catabolic genes	Amino acid metabolism (Y) Pol II transcription (S,Y)	Amino acid biosynthesis (e^{-06},8) Urea cycle intermediate metabolism ($5.4e^{-05}$,4) Nitrogen metabolism ($7.8e^{-05}$,4)	**Success**
Apg1 (168) Signaling	Serine/threonine protein kinase involved in induction of autophagy after nutrient limitation	Autophagy (S) Meiosis (Y) Protein degradation (Y) Vesicular transport (Y)	Protein biosynthesis (0,29) Structural protein of ribosome (0,24) Hydrolase, hydrolyzing O-glycosyl compounds (0.000288,6)	**Partial Success**
Slt2 (245) Signaling	Serine/threonine (MAP) kinase involved in the cell wall integrity (low-osmolarity) pathway.	Signal transduction (S,Y) Cell stress (Y) Cell wall maintenance (Y) Protein amino acid phosphorylation (S)	Cell wall structural protein ($4e^{-06}$,6) Cell wall organization and biogenesis ($5.4e^{-05}$,11) Protein kinase cascade (0.017959,2)	**Success**
Tpk1 (603) Signaling	Catalytic subunit of cAMP-dependent protein kinase 1, protein kinase A or PKA.	Signal transduction (Y) Aging (Y) RAS protein signal transduction (S) Pseudohyphal growth (S)	Ribosome biogenesis ($7e^{-06}$,34) Protein biosynthesis ($3.3e^{-05}$,58) Structural protein of ribosome (0.000128,46)	**Partial Success**
Sip4 (265) TF	Transcriptional activator of gluconeogenic genes through CSRE elements.	Transcription factor (S,Y)	Structural protein of ribosome ($1.1e^{-05}$,28) Protein biosynthesis ($6.9e^{-05}$,31)	**No support**
Tec1 (104) TF	Transcriptional activator, involved with Ste12p in pseudohyphal formation	Differentiation (Y) Pseudohyphal growth (S)	Mating ($3.2e^{-05}$,9) Pheromone response ($9.2e^{-05}$,6) Cell surface receptor linked signal transduction (0.013253,3)	**Partial Success**

Table 5.1: Functional annotation of top-10 active regulators sorted by p-value of most significant GO term. For each active regulator, an existing annotation (adapted from YPD) and known GO annotation (from SGD and YPD) are compared to significant GO terms within its regulatee set (derived GO terms). Five of the derived annotation fully match known ones (Success), three match part of the known annotation (Partial Success), one is a novel functional assignment for a previously uncharacterized TF (novel function) and one is completely inconsistent with known functions (no support).

Regulator (# regulates)	Concise annotation (YPD)	Significant GO terms (score, # genes)	Suggested novel annotation
Tea1 (141) TF	Ty1 enhancer activator of the Gal4p-type family of DNA-binding proteins	Amino acid metabolism (0,15) Nitrogen metabolism ($7.9e^{-05}$,5) Urea cycle intermediate metabolism (0.000679,4)	**Regulation of amino acid biosynthesis (nitrogen utilization?)**
Kin1 (292) Sig	Serine/threonine kinase. Null mutant is viable and **shows no obvious phenotype**	Protein biosynthesis (0.000182,32) Cytokinesis (0.003699,6) Budding (0.005838,8) GTPase activator (0.019626,3) Cell cycle (0.034484,18)	**Cell cycle regulation (budding and cytokinesis)**
Yol128C (170) Sig	Protein kinase. **Unknown biological process and molecular function.**	Cell communication (0.000362,14) Nitrogen starvation response (0.000944,3) Response to external stimulus (0.00426,8) Cell cycle control (0.020422,5) Cell cycle (0.031847,12) Signal transduction (0.034232,6)	**Cell cycle regulation, perhaps in response to certain starvation signals**
Kin82 (127) Sig	Putative serine/threonine protein kinase. **Biological process unknown.**	Nucleotide metabolism (0.002014,3) Primary active transporter (0.003399,6) Mating-type specific transcriptional control (0.008909,2)	**Potential participant in the mating response pathway**

Table 5.2: Novel functional assignment for previously uncharacterized regulators with the MinReg algorithm. For each active regulator, significant GO terms were derived based on their regulatee set. These served as a basis for functional annotation of the regulator. In some cases (e.g. YOL128C, see text), the annotation was independently supported by external data.

the molecular function of the effectors (cell wall structural proteins and endopeptidases), the molecular mechanism of regulation (protein kinase cascade) and associated, cross-talking modules (mating, cell-cell fusion). Inspection of individual regulatee genes further supported this annotation[4]. Most importantly, MinReg found Slt2 to regulate Rlm1, the main transcription factor of the low-osmolarity cell-wall integrity pathway. Rlm1 is phosphorylated and activated *post-transcriptionally* by Slt2 (an event that cannot be observed in expression data), and, in turn, directly mediates Slt2's activatory effect. Indeed, several of the aforementioned regulatees (Pst1, Crh1, Bop1, Ktr2, Gsc2, Yps3, Prp2) are known Rlm1 targets and promoter sequence analysis of Slt2's regulatees indicated a highly significant Rlm1 motif ($P = 2.85e^{-05}$).

Our method can also correctly piece together the joint regulation of a specific biological process by several active regulators. For example, we detected several GATA transcription factors (Gat1, Uga3, Dal80) that are known to participate in the regulation of the nitrogen starvation response (see Figure 5.4). All three regulators were associated with correct biological processes ("nitrogen starvation response" for Gat1 and Dal80, "urea cycle" and "nitrogen metabolism" for Uga3). However, the molecular functions they regulate only partially overlapped, highlighting their specific roles in a common response. Thus, only Dal80 regulates allantoin pathway genes (Dal1,2,3 and 7, consistent

[4]Among the regulatees are known and putative structural cell wall proteins (Cis3, Sed1, Tir1, Pir3, Pir1, Bop1, Pau1, Pau6, Pry2, Dan1, Dan3, Tir1, Crh1, Pst1), cell wall enzymes (Asp3-2, Bgl2), cell wall aspartic endopeptidases (Yps3, Yps5, and Yps6), enzymes and other genes involved in cell wall biogenesis (Myo3, Krt2, Krt6, Gsc2, Chs1) and transcriptional regulators known to affect cell wall maintenance (Rlm1, Ecm22, Tup1). Many of these genes were previously reported to be transcriptionally regulated by various stress conditions, such as heat- and cold-shock or changes in osmolarity. Most of the identified mating and cell-cell fusion genes (Aaf1, Cdc1, Prm8, Prm5, Prm10, Cmp2, Scw10, Prp2, Gic2) are cell-wall related through biological process (budding, stress) or localization (membrane or cell wall proteins).

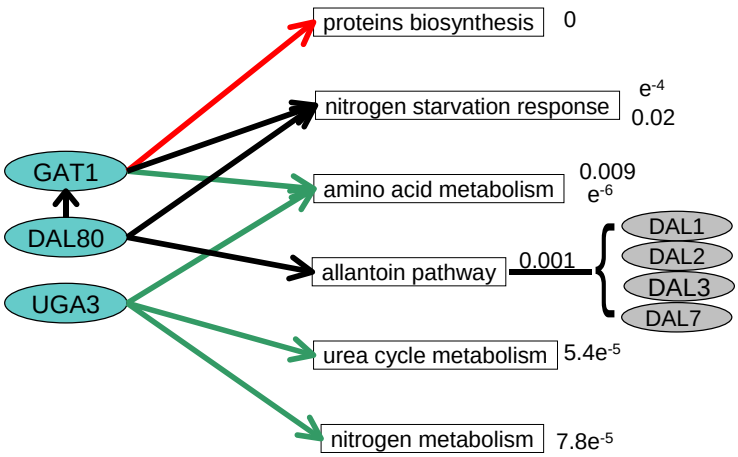

Figure 5.4: Our reconstruction of GATA regulation. The arrows represent associations between regulators and the process they regulate. Green arrows represent activation and red arrows represent repression. The p-value of each association is noted to the left of each process.

with the specific effect of dal80-null mutant on allantoin catabolism [15]), while only Gat1 regulates protein biosynthesis and ribosome biogenesis, another molecular component of the response. Our model also correctly captured *inter-regulator links*: we found that Dal80 regulates Gat1, in accordance with recent findings on exactly such direct transcriptional regulation [23].

In certain cases, our methodology allows us to assign function to previously uncharacterized regulators (Table 5.2), or to expand the functional assignment of known ones. For example, it associated the protein kinase YOL128C with the GO terms “Nitrogen starvation response”, “response to external stimulus”, and “cell cycle control”. Among Yol128C’s specific regulatees we find numerous cell cycle genes and regulators[5]. Thus, we postulate that YOL128C acts as a cell cycle regulator in response to starvation signals. This prediction is further supported by Yol128C’s homology to the meiosis regulator Gsk3.

For known regulators, our method may refine or expand existing annotations. For example, Xbp1, a transcriptional repressor induced by stress and starvation during the mitotic cycle, was assigned both with a “stress response” term and with a “sporulation” one. The latter term suggests an extension of Xbp1’s role to sporulation. In fact, this postulated function is consistent with a recent report of Xbp1’s role in efficient sporulation [66].

In other cases, our functional assignment may be misleading. For example, Apg1, a signaling molecule involved in induction of autophagy after nutrient limitation, is strongly associated with “protein biosynthesis” and “structural proteins of the ribosome”. Although Apg1 is not known to regulate this process, Apg1 is regulated by TOR proteins, which are known to regulate *both* ribosome biogenesis and autophagy [78]. Tor1 itself is regulated post-transcriptionally, as reflected by its unchanged expression in most of the samples. In the absence of a TOR signal in the data, we are

[5] e.g. Rsc3, Cdc22, Cdc25, Net1, Kcc4, Nip29, Dmc1, Ulp1, Pcl6, Tem1, Elm1, Chs3, Spa2

capturing Apg1 as its "replacement" in our model, reflecting co-regulation rather than true regulation. Another Tor1 target TF, Gat1, shows a similarly strong association to "protein biosynthesis" for the same reason. Note, that Apg1's true role in autophagy is supported by other GO terms, including O-glycosyl hydrolases, as well as by specific autophagy genes (Aut2, Aut4) in its regulatee set.

The regulation logic we unraveled was often consistent with that reported in the literature. For example, we found that Met28 activates the biological processes of "threonine and methionine amino acid metabolism" as well as "sulfur utilization". Similarly our method correctly predicted the activatory roles of Slt2 and Uga3. All these findings are consistent with the known roles of these regulators (see table 5.2). Importantly, the same regulator may assume different logical roles with respect to different processes or functions. For example, we found that Gat1 activates several amino acid metabolic processes, but inhibits "protein biosynthesis" and "ribosome biogenesis". This is fully consistent with Gat1's known role in mediating the nitrogen starvation response, which involves both increase in amino acid metabolism and concomitant transcriptional down-regulation of ribosomal proteins and other biosynthetic genes [61].

Our derived logic is particularly useful for the annotation of previously uncharacterized proteins. For example, we found that Yol128C positively regulates the "nitrogen starvation response", while negatively regulating "cell cycle control". This indicates that the gene may regulate a stress response by inhibiting cell cycle progression under starvation conditions.

Regulatory logic must be interpreted with care, especially when the regulator is a signaling molecule. For example, we found that Sst2 positively regulates mating and transmembrane receptors, while in fact it is a negative regulator of both (Table 5.5). To explain this discrepancy, note that Sst2 is activated by Ste12, the main mating TF, along with most of the mating pathway genes, while it exerts its inhibitory effect on the pathway only later, after a time delay. Thus, rather than representing its own (negative) regulatory role, Sst2 serves as a representative of its fellow, co-activated mating genes. We believe that in such cases (mostly involving signalling molecules), we cannot reconstruct the regulatory logic from steady-state expression profiles. Nevertheless, we conclude that correct logic may often be derived by our analysis.

5.6 Systematic Evaluation

5.6.1 Robustness and Cross Validation

One of the main concerns when using only a few hundred samples to build a model over thousands of variables is the statistical robustness of the resulting model. Such scenarios often lead to significant overfitting, resulting in model features that correspond to spurious signal. We would like to ensure that our learned model indeed corresponds to real biological signal. In Section 5.5, we provided evidence of such a correspondence by demonstrating that automated annotations derived from our model successfully match the known biological role of the inferred regulators. This section offers complimentary statistical evidence: We ask how well does the model generalize to unseen data?

Recall, in our model, each $X \in \mathcal{X}$ is associated with a probabilistic function of its regulators (i.e. each instantiation $\mathbf{pa}_X$ defines a distribution $P(X \mid \mathbf{pa}_X)$). We use the empirical counts observed in the data to estimate these distributions.

$$P(X = x \mid \mathbf{Pa}_X = \mathbf{pa}_X) = \frac{M[X = x, \mathbf{Pa}_X = \mathbf{pa}_X]}{M[\mathbf{Pa}_X = \mathbf{pa}_X]}$$

We use these distributions to evaluate the regulator's ability to predict the expression of its regulatees. Recall that our model contains only a small number (tens) of regulators. Given the expression levels of only these regulators, we use the model and the probabilities $P(X \mid \mathbf{Pa}_X)$ to "predict" the expression levels of all other variables. Given a data sample in which $\mathbf{Pa}_X = \mathbf{pa}_X$, our model "guesses" X by randomly sampling from $P(X \mid \mathbf{pa}_X)$.

A simple minded test would be to use MinReg to infer a model and then measure how often this model correctly predicts each variable. Given the expression of regulators in the mth sample, we use $P(X \mid \mathbf{pa}_X[m])$ to predict the value of X in that sample. For each X, we measure how well we can predict X by calculating :

$$\frac{1}{M} \sum_{m=1}^{M} P(x[m] \mid \mathbf{pa}_X[m])$$

However, it is not an impressive feat to correctly predict the same samples used by the algorithm to train the model. A learning procedure can easily fit a model to the data. The resulting model would attain excellent performance on the training data, but would perform poorly on samples not encountered during the training process.

Our goal is to learn the underlying process that created that data, not the data itself. The real test is: Can we use our model to correctly predict gene expression of a new sample How can we generate such a new microarray sample? The standard trick is called *cross validation*: randomly partition the data samples into two disjoint sets, a training set and a test set. The MinReg algorithm sees only the training set when learning the model and then uses only test set to measure its performance.

We evaluated the performance of our algorithm using 5-fold cross validation. The arrays (data samples) were randomly partitioned into five equal sized sets. We ran the MinReg algorithm 5 times choosing a different test set each time. In each run, MinReg removed the test set from its input and inferred a model using only the training data. For each test sample, the values of all 3755 regulatees (genes) were predicted using the inferred model. We calculated the overall precision of the model by comparing our predicted values with the actual measured ones.

We compared the predictive capabilities of several different models. As a baseline, we used the marginal probability of each gene to predict its value. Since most of the genes remained unchanged most of the time, even this simple predictor scored well (Figure 5.5, crosses). As competition to our MinReg algorithm, we generated 45 clusters using standard k-means clustering [28] and randomly chose from within each cluster a gene $r \in \mathcal{C}$ as its "regulator". For each cluster we calculated $P(X \mid r)$ and used this as our predictor. While cluster representatives somewhat improved the

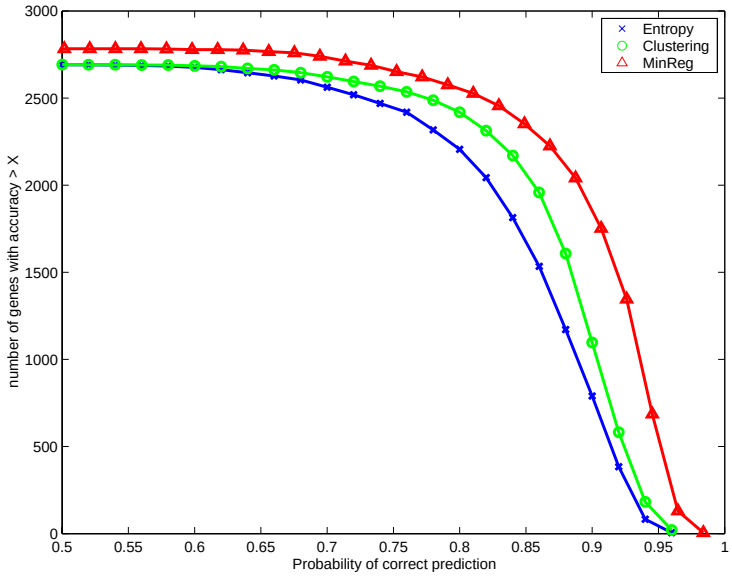

Figure 5.5: Cross validation of the predictive capabilities of our model on test data. The graph measures the number of genes correctly predicted at each probability. We compare our model (triangles) to the null model (crosses) that uses the marginal distribution of each gene and to a model based on cluster representatives (circles)

prediction (0.06 log-loss/instance, Figure 5.5, circles), our MinReg algorithm clearly provided the best predictions (0.11 log-loss/instance, Figure 5.5, triangles).

Several important advantages of MinReg account for its predictive success over clustering. These include capturing "mechanistic" regulation using mutual information (vs simpler co-expression in clustering), identifying both activatory (correlated) and inhibitory (anti-correlated) regulators, and finding combinatorial logic and non-linear interactions between a gene and its regulators (impossible in clustering where the same gene cannot belong to more than one cluster). In conclusion, our cross-validation demonstrates that most of the information in an entire array can be captured by a small set of master predictors (the regulators chosen by MinReg).

5.6.2 The Importance of Candidate Regulators

Supplying the MinReg algorithm with prior knowledge of potential regulators is a potent supplement to our algorithm. In addition to focusing the algorithm onto regulatory relations, it narrows the search space and significantly reduces the running time of the algorithm (which is quadratic in the size of candidate set). In this section we assess how much this prior knowledge contributes to our algorithm's success. We ask: can the MinReg algorithm correctly reconstruct regulation without an apriori set of candidate regulators?

To answer this question we withdrew the candidate regulator set and ran the MinReg algorithm so that $\mathcal{C} = \mathcal{X}$, i.e., any gene could be chosen as a regulator. Since the algorithm is quadratic in $\mathcal{C}$, we reduced $\mathcal{X}$ by including only genes whose expression significantly changed in ≥ 18 samples

(verses 15 samples). This resulted in a set of 2828 genes for both $\mathcal{X}$ and $\mathcal{C}$. It is important to note that in our data set, the expression of many candidate regulators remains almost constant. Only 148 genes from our original candidate set of known and putative regulators were included in the 2828 genes.

The MinReg algorithm chose 35 regulators, of which only 6 genes (Ste2,Spl2,Wtm1,Rpi1,Bas1, Phd1) are in the original candidate regulator set. We speculate that this is because the behavior of a co-regulated gene is often at least as predictive (and sometimes even more) than the behavior of the regulating gene itself. Therefore, when the candidate regulator set is not limited to real regulators, most of the regulators chosen by the MinReg algorithm do not have a regulatory molecular function. While this model might be highly predictive and even generalizes well to new data, it does not fulfill our goal to reconstruct biological regulation. Furthermore, it is difficult to assign a biological interpretation to such a model.

Nevertheless, while $\frac{6}{35}$ is only a small fraction of the chosen regulators, this is a significant enrichment compared to their fraction in the candidate set (pvalue 0.003). The fact that our procedure results in a statistically significant enhancement of regulators is encouraging. We speculate that in complex organisms, where combinatorial regulation most likely plays a stronger role, this approach will be even more successful in detecting genes with a regulatory function.

5.6.3 Simulated Data

We evaluated MinReg on synthetic data. We used the regulation model generated from the data as our synthetic network (the network contained 2828 genes and 44 regulators). We randomly sampled 10 datasets from this network, each set consisting of 358 samples (same number of samples as the original dataset). We ran MinReg on each of these 10 synthetic datasets using the same parameters used in Section 5.5.

Our first test was more global. We evaluated MinReg's choice of regulators. In average MinReg correctly reconstructed 84% (39) of the generating 44 regulators. The worse case was 80% (35) and best case 91% (40). As for false positives, in average 74% of the reconstructed regulators were correct (worse case 71% and best case 77%).

Next we evaluated the detailed model itself. The generating model contained 6616 individual regulatory relations and we checked how many of those were recovered. In average 70% of these were recovered (worse case 69% and best case 72%). In each model the percent of correct relationships was a bit higher, with an average of 74% (worse case 72% and best case 77%).

Even when we did not limit to candidate regulators (setting $\mathcal{C} = \mathcal{X}$) our reconstruction of regulators was surprisingly good. Using the same parameters as before 42/92 of the regulators were correct. The earlier iterations were most accurate, when using a stricter stopping criteria, 42/47 of the regulators were correct. When using the stricter stopping criteria, %76 of the individual edges were correctly reconstructed.

To summarize using only a small number of samples MinReg is capable of learning a model over thousands of genes, correctly reconstructing most of the relationships.

5.7 Discussion

In this chapter we presented an efficient novel algorithm for inferring regulatory relationships. Our learning succeeds in two challenging tasks:

- We succeed in predicting significant portions of a microarray based solely the values of a small number of genes.
- We automatically associate a comprehensive functional annotation to regulators.

Our detailed analysis shows that most of our high scoring assignments are correct, and that novel roles can be assigned to previously uncharacterized transcription factors and signaling proteins.

Our proposed method lies between molecular network modeling [36, 76, 93] and global methods for the analysis of gene expression (e.g. clustering [30, 86] and PCA [4, 50]), and offers a number of advantages over both approaches. On the one hand, unlike previous network modeling approaches, MinReg can robustly produce a biologically significant regulation network over *thousands* of genes, thus allowing it to scale to mammalian genomes. On the other hand, the regulation structure we learn goes beyond clustering and PCA by capturing combinatorial logic and non-linear behavior, both of which are important in biological systems. Indeed, MinReg's predictive superiority over clustering was clearly demonstrated.

The main drawback of the MinReg approach is that it does not reconstruct a detailed regulation network, but rather a coarse abstract model of regulation. Our small regulator set only contains regulators which have a broad affect in data, possibly missing many regulators that act on a smaller scale. Furthermore, regulatory relations in our models represent both direct and indirect regulation with no clear distinction between them. We note that it is this abstraction which allows MinReg to robustly reconstruct regulation over thousands of genes from so few data samples.

Furthermore, our model should to be interpreted only at the more global scale. We do not have high confidence in any single regulatory relation in the model as some of these are spurious. Rather, we have confidence in the global properties of regulator sets: regulator r regulates process X verses of regulator r regulates gene X. We note that if r is associated with process P and X is annotated with P, than the edge $r \rightarrow X$ is more likely to represent a true relationship.

Chapter 6

Module Networks - Reconstructing Regulatory Modules

The complex functions of a living cell are carried out through the concerted activity of many genes and gene products. This activity is often coordinated by the organization of the genome into regulatory modules, or sets of co-regulated genes that achieve a common function. Such is the case for most of the metabolic pathways as well as for members of multi-protein complexes. Revealing this organization is essential for understanding cellular responses to internal and external signals. Clustering allows the identification of groups of co-expressed genes. However, the regulatory programs of these groups are only indirectly suggested by finding common *cis*-regulatory binding sites in the upstream regions of genes within each group [90, 94].

In Chapter 3, we inferred regulation based on the statistical dependencies between regulators and targets. This reconstruction is based on the key assumption that the regulators are themselves transcriptionally regulated, so their expression profiles provide evidence as to their activity level. In Chapter 5, we focus the learning on regulation by providing the algorithm with a list of genes that potentially act as regulators and restrict the regulators in our models to these sets. By enforcing a small number of shared regulators, our model gained in statistical robustness. In this chapter, we take this idea one step further: genes will not only share regulators, they also share regulatory programs. We propose Module Networks: a unified probabilistic framework that combines the best of both clustering and network modeling. Our procedure identifies modules of co-regulated genes, their regulators, and the shared regulation program that governs their behavior.

In Section 6.1 we describe the *Module Networks* framework. In Section 6.2 we explain how Module Networks can be used to reconstruct regulatory modules from gene expression data. Section 6.3 presents a comprehensive biological analysis of the inferred module networks. Section 6.4.2 and Section 6.5 provide technical details of formal definitions, scoring functions and algorithms. Finally Section 6.6 offers a systematic evaluation of the algorithm's performance.

6.1 From Bayesian Network to Module Network

Genes required for the same biological function or response are often co-regulated in order to coordinate their joint activity. One such example is Slt2, a MAP kinase which activates the genes of the cell wall integrity low osmolarity response pathway. In this case, Slt2 exerts a similar regulatory affect on each of its targets. A Slt2 *hub* was automatically inferred by our Bayesian network algorithm. The learning algorithm inferred a parent-child relation between Slt2 and its targets, independently for each target, forming the subnetwork presented in Figure 3.5.3. The Bayesian network model did not explicitly model Slt2 as a regulator of an entire response.

An Slt2 hub was also inferred by the MinReg algorithm (Section 5.5). While MinReg explicitly searches for regulators that are pronounced in the data and regulate an entire ensemble of genes, the algorithm does not take the coordinated nature of the response into account. For each target, both the fact that Slt2 regulates it and Slt2's regulatory affect was inferred in an independent manner. We interested in a computational model that automatically organizes Slt2's targets into a *regulatory module* and enforces a common *regulatory program* on all its targets.

We define a new representation called a *module network*, which explicitly partitions the variables (genes) into *modules*. Each module represents a set of variables that have the same statistical behavior, i.e., they share the same set of parents and local probabilistic model. By enforcing this constraint on the learned network, we significantly reduce the complexity of our model space as well as the number of parameters.

A module network consists of two components.

1. A component that defines a template probabilistic model shared by all the variables assigned to that model. This template includes both a set of regulating parents and the conditional probability distribution they specify over the module genes.
2. A module assignment function that assigns each variable to one of the modules. A module network can be viewed simply as a Bayesian network in which variables in the same module share parents and parameters (see Figure 6.1).

Given the small number of samples in typical gene expression datasets, noise in the data makes it extremely unlikely that such a modular structure would arise naturally from a Bayesian network learning algorithm, even if it exists in the domain. Moreover, by making the modular structure explicit, the module network representation provides insights about the domain that are often be obscured by the intricate details of a large Bayesian network structure. It would be very difficult to intepret a Bayesian network on a genome-wide scale because of its intricate circuitry over thousands of genes.

We note that the true biological network is not a Module network. While genes in the same module might have similar regulatory control, they do not share the exact same set of regulators and regulatory program. Nevertheless, using module networks as a first order approximation can lead to a more robust estimation and better generalization of unseen data.

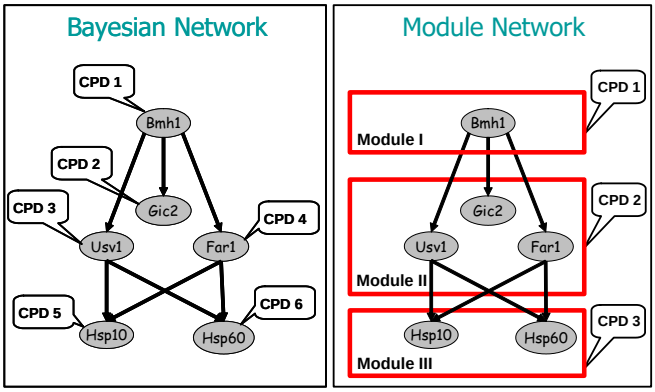

Figure 6.1: (a) A simple Bayesian network over gene expression variables. (b) A simple module network; the boxes illustrate modules, whose genes share CPDs and parameters.

There are several reasons why a learned module network is a better model than a learned Bayesian network. Most obviously, parameter sharing between variables in the same module allows each parameter to be estimated based on a much larger sample. Moreover, this allows us to learn dependencies that are considered too weak based on statistics of single variables. These are well-known advantages of parameter sharing; but a novel aspect of our method is that we automatically determine which variables have shared parameters.

Furthermore, the assumption of shared structure significantly restricts the space of possible dependency structures, allowing us to learn more robust models than those learned in a classical Bayesian network setting. While the variables in the same module might behave according to the same model in underlying distribution, this will often not be the case in the empirical distribution based on a limited number of samples. A Bayesian network learning algorithm will treat each variable separately, optimizing the parent set and CPD for each variable in an independent manner. In the very high-dimensional domains in which we are interested, there are bound to be spurious correlations that arise from sampling noise, inducing the algorithm to choose parent sets that do not reflect real dependencies, and will not generalize to unseen data. Conversely, in a module network setting, a spurious correlation would have to arise between a possible parent and a large number of other variables before the algorithm would find it worthwhile to introduce the dependency.

6.2 From Module Network to Regulatory Module

In this section we model of gene regulation using the module network framework. The key link between our gene expression domain and the module network model is the *regulation program*. The regulation program of a module specifies the set of regulatory genes that control the module and the mRNA expression profile of the genes in the module as a function of the module's regulators.

These regulation programs correspond to the conditional probability distributions. Contrary

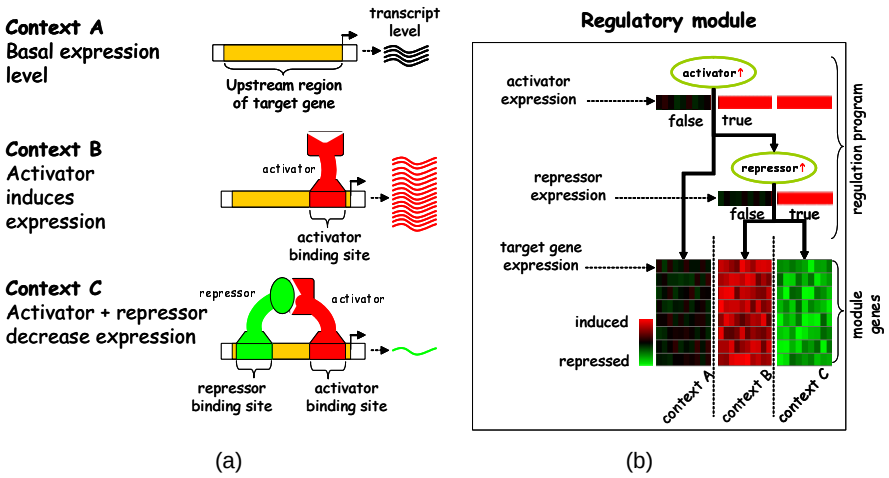

Figure 6.2: (a) Cartoon depicting three distinct modes of regulation for a group of genes. (b) Regulation tree that represents these modes of regulation. The expression of the regulatory genes is shown below their respective node. Each leaf of the regulation tree is a regulation context (bordered by dotted lines) as defined by the queries leading to the leaf. The arrays are sorted each into their respective context and the expression of the regulatory genes themselves is shown: each row corresponds to a gene and each column to an array.

to previous chapters, in this chapter, we bestow CPDs with both structure and meaning. These regulation programs are represented using a *regression tree* [9]. Regression trees were first used to model gene expression by Segal et. al [83]. They [83, 80] identify modules of co-regulated genes and their shared *cis*-regulatory motifs, but do not suggest regulator genes nor regulatory programs. In contrast, our method discovers both regulatory modules and their control programs, suggesting concrete regulators for each module, their effect and combinatorial interactions.

The following example illustrates how a regression tree models biological regulation. Consider a group of genes that are all regulated by the same combination of activator and repressor genes, resulting in three distinct modes of regulation (see Figure 6.2 (left)). In context A, the genes in the module are not under transcriptional regulation and are in their basal expression level. In context B, the activator gene is up-regulated and as a result binds the upstream regions of the module genes, thereby inducing their transcription. In context C, a repressor gene is also up-regulated and as a result blocks transcription of the genes in the module, thereby decreasing their expression levels.

A regulation program can represent the module's response to the different regulatory *contexts*. A *context* is a rule describing the qualitative behavior (up-regulation, no change or down-regulation) of a small set of genes that control the expression of the genes in the module. These rules are organized as a tree, where each path to a leaf in the tree defines a context via the tests on the path. This tree is composed of two basic building blocks: decision nodes and leaf nodes. Each decision node corresponds to one of the regulatory inputs and a query on its value (e.g., "is Hap4 up-regulated?"). Each decision node has two child nodes: the right child node is chosen when the answer to the corresponding query is true, the left node is chosen when not. For a given array, one begins at the

root node and traverses down the tree in a path depending on the answers to the queries in that particular array, until a leaf is reached. We use Figure 6.2 (right) to illustrate the path for context B: The root node's query is "Is the activator up-regulated?", in context B this is true so we continue to the right child. This node contains the query "Is the inhibitor up-regulated?", in context B this is false so we take the left child. The resulting leaf corresponds to a regulation context in which the genes are over-expressed, which is indeed the behavior of the module genes in context B.

Each leaf of the regulation tree is a context that specifies the behavior of a set of arrays: those in which the tree traversal reaches the corresponding leaf. The response in each context is modeled as a normal distribution over the expression values of the module's genes in these arrays; this distribution is encoded using a mean and variance stored at the corresponding leaf. The model semantics is: given a gene g in the module and an array a in a context, the probability of observing some expression value for gene g in array a is governed by the normal distribution specified for that context. For each array, all genes in the same module follow the same normal distribution. In a context where the genes are tightly co-regulated, the distribution will have a small variance. In a context where the genes are not tightly regulated, the distribution may have a large variance. Thus, a regression tree allows for expression profiles with different degrees of conservation of the mean behavior of the module.

This notion of a regulatory program has several key advantages:

- It captures combinatorial interactions; e.g., the module is strongly up-regulated if both Hap4 and Alpha2 are up-regulated.
- It handles context specific effects and environmental conditions; e.g., Hap4 regulates the module only in presence of respiratory carbon sources.
- It allows modules that have a conserved signature only in certain contexts; e.g., a module may have strong coordinated response to Gat1 up-regulation, yet have diffuse behavior in other settings.

6.2.1 Algorithmic Overview

We propose a fully automated procedure that discovers such modules directly from gene expression profiles. For clarity, we present a very simplified overview that captures the algorithm's essence (more details are given in Section 6.4.2 and Section 6.5). Given a gene expression data set and a precompiled set of candidate regulatory genes, the algorithm simultaneously searches for a partition of genes into modules, and for a regulation program for each module that explains the behavior of the genes in the module. Our algorithm takes an iterative approach. We begin with an initial clustering of the genes based on their expression profiles. For each cluster of genes, the procedure searches for a regulation program that provides the best prediction of the expression profiles of genes in the module. After identifying regulation programs for all clusters, the algorithm re-assigns each gene to the cluster whose program best predicts that gene's behavior. The algorithm iterates

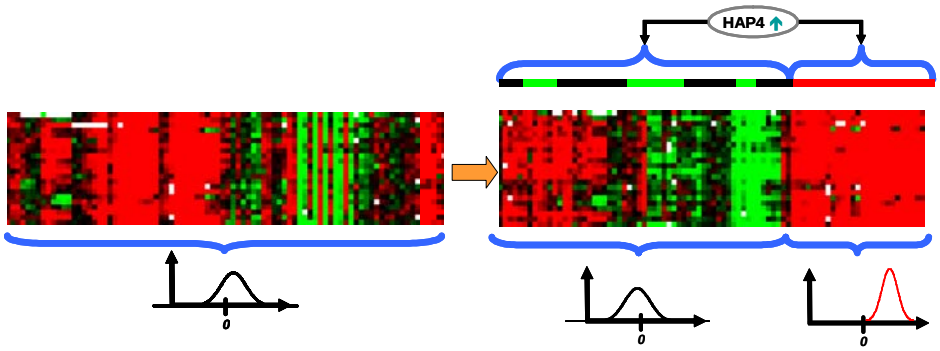

Figure 6.3: Search step in the regression tree learning algorithm. To the left the arrays are in no particular order. The query "is Hap4 upregulated" partitions the arrays into two distinct distributions (right). In arrays in which Hap4 is over-expressed the module genes are strongly up-regulated. The distribution for each of the two leaves is distinctly different.

until convergence, refining both the regulation program and the gene partition in each iteration. The procedure outputs a list of modules and associated regulation programs.

The regulation program is learned via a combinatorial search over the space of trees. The tree is grown from the root to its leaves. At any given node, the query which best partitions the gene expression into two distinct distributions is chosen, until no such split exists. Figure 6.3 illustrates how such a query can partition the arrays into two different modes of regulation. A good partition is one that results in two distinct distributions. In our example, the module genes are all strongly upregulated in arrays in which Hap4 is upregulated.

Given the inferred regulation programs, we determine the module whose associated regulation program best predicts each gene's behavior. Recall, each regulation program defines a probability distribution over the gene's expression levels in each array. We test the probability of a gene's empirical expression values in the dataset under each regulatory program as follows: for each array, we evaluate the probability of the associated expression measurement in that array's context as specified by the regression tree; we then multiply the probabilities for the different arrays, obtaining an overall probability that this gene's expression profile was generated by this regulation program. We then select the module whose program gives the highest probability, and re-assign the gene to this module. We take care not to assign a regulator gene to a module in which it is also a regulatory input, since it is not surprising that a gene can predict its own gene expression.

Our two-step iterative learning procedure attempts to search for the model with the highest score, in each step optimizing one of the models components: regulation programs or gene partition. An important property of our algorithm is that each iteration is guaranteed to improve the likelihood of the model, until convergence to a local maximum of the score.

6.3 Biological Results

We used an *S. cerevisiae* gene expression dataset consisting of 173 microarrays that measure responses to various stress conditions [40]. We chose a subset of 2355 genes which display a significant change in gene expression under the measured stress conditions, excluding members of the generic environmental stress response cluster [40]. We used the set of 466 genes described in Section 5.1 as our candidate regulators. We initialized our module network procedure with 50 clusters using PCluster, a hierarchical agglomerative clustering (see Section 6.5.2). We then applied the iterative learning algorithm to refine both the gene partition and the regulatory programs. Our procedure converged after 23 iterations[1] resulting in the 50 modules we analyzed.

With few exceptions, each of the inferred modules (46/50) contained a functionally coherent set of genes. Together the modules spanned a wide variety of biological processes including metabolic pathways (e.g., glycolysis), various stress responses (e.g., oxidative stress), cell-cycle related processes, molecular functions (e.g., protein folding), and cellular compartments (e.g., nucleus) (see [82] for complete listing of modules). Most modules (30/50) included genes previously known to be regulated by on of the module's predicted regulators. Many modules (15/50) had a match between a predicted regulator and its known *cis*-regulatory binding motif (i.e., a statistically significant number of the module's genes contained the known motif in their upstream regions). Overall, our results provide a global view of the yeast transcriptional network, including many instances in which our method discovers known functional modules and their correct regulators, demonstrating the ability of our method to derive regulation from expression.

6.3.1 Selected Modules

We first present in detail several of the inferred modules, selected to show the method's ability to recover diverse features of regulatory programs.

The Respiration Module (see Figure 6.4) provides a clear demonstration of a predicted module. It consists primarily of genes encoding respiration proteins (39/55) and glucose metabolism regulators (6/55). The inferred regulatory program specifies the Hap4 transcription factor as the module's top (activating) regulator, primarily under stationary phase conditions, consistent with Hap4's known role in activation of respiration [26]. Indeed, our post-analysis detected a Hap4-binding DNA sequence motif (bound by the Hap2/3/4/5 complex) in the upstream region of 29/55 genes in the module ($p < 2 \times 10^{-13}$). Note that this motif also appears in non-respiration genes (mitochondrial genes and glucose metabolism regulators), which together with their matching expression profiles, supports their inclusion as part of the module. When Hap4 is not induced, the module is activated more mildly or is repressed. The method suggests that these changes are regulated by other regulators, such as the protein phosphatase type 1 regulatory subunit Gac1 and the transcription factor Msn4. Indeed, the stress response element (STRE), recognized by Msn4, appears in the upstream

[1] For computational efficiency most iterations only optimize the parameters of the normal distributions at the leaves and leave the tree structure unchanged. For the reported results, only four tree structure change iterations were applied

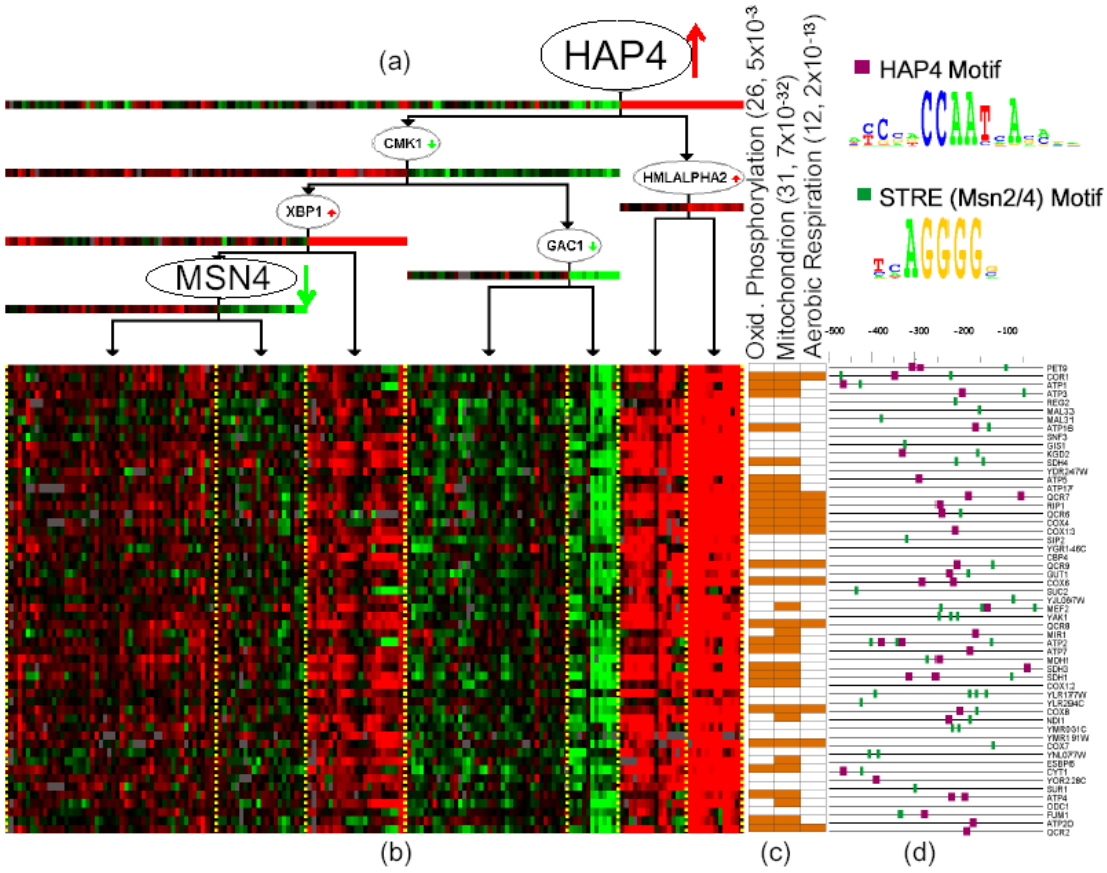

Figure 6.4: The Respiration and Carbon Regulation Module. (a) Regulation tree/program: The expression of the regulators themselves is shown below their respective node. (b) Gene expression profiles: Genes (rows), arrays (columns). Arrays are arranged according to the regulation tree. For example, the rightmost leaf includes the arrays in which both Hap4 and Alpha2 are up-regulated. (c) Significant annotations: Colored entries indicate genes with the respective annotation. (d) Promoter analysis: Lines represent 500bp of genomic sequence located upstream to the start codon of each of the genes; colored boxes represent the presence of *cis*-regulatory motifs located in these regions. Note, the enrichment of both the HAP4 motif (purple) and the stress response element (STRE; green), recognized by Hap4 and Msn4, respectively

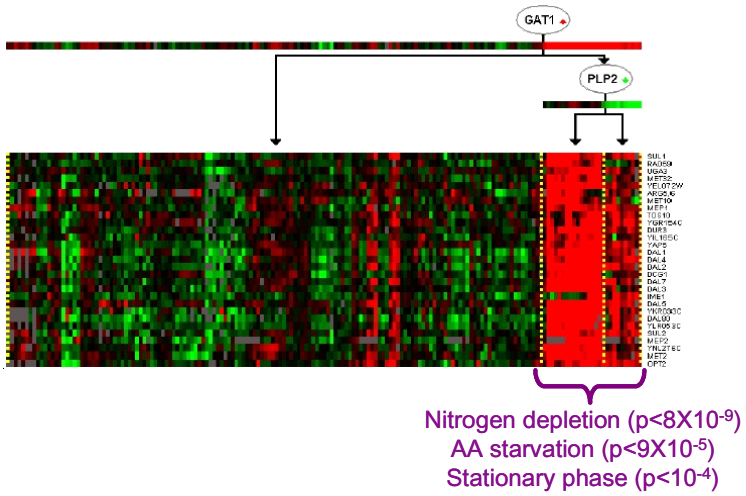

Figure 6.5: The Nitrogen Catabolite Repression Module demonstrates a *Module signature*: tight co-expression (assumed to be co-regulation) around a specific sub-set of conditions

region of 32/55 genes in the module ($p < 10^{-3}$) as well as in those of many of the genes containing the Hap4 motif (17/29 genes; $p < 7 \times 10^{-10}$), supporting our placement of both regulators in one control program.

The Nitrogen Catabolite Repression Module (see Figure 6.5) demonstrates the ability of our method to capture an entire cellular response whose genes participate in diverse metabolic pathways and cellular roles (12/29 in allantoin and urea metabolism, 5/29 in amino acid metabolism, and 6/29 in sulfur/methionine metabolism), all of which relate to the process by which the yeast utilizes the best available nitrogen source. Gat1 is suggested as the key (activating) regulator of this module, further supported by the presence of the GATA motif, the known binding sequence for Gat1, in the upstream region of 26/29 genes ($p < 10^{-17}$). This module also demonstrates that the method can discover context-specific regulation, as the similarity in expression of genes in the module is mostly pronounced in stationary phase (17/22 experiments; $p < 10^{-4}$), amino acid starvation (5/5; $p < 9 \times 10^{-5}$), and nitrogen depletion (10/10; $p < 8 \times 10^{-9}$), all of which are conditions where nitrogen catabolism is active. Note that two additional known regulators involved in this response, Uga3 and Dal80, are suggested as members, rather than regulators, of the module.

The Energy, Osmolarity and cAMP Signaling Module demonstrates that our method can discover regulation by proteins other than transcription factors, as the top predicted regulator was Tpk1, a catalytic subunit of the cAMP dependent protein kinase (PKA). This prediction is supported by a recent study [71] showing that expression of several genes in the module (e.g., Tps1) is strongly affected by Tpk1 activity in osmotic stress, which was among the conditions predicted by the method to be Tpk1-regulated. Further support is given by the presence of the STRE motif, known to be bound by transcription factors that are regulated by Tpk120, in the upstream region of most genes in the module (50/64; $p < 3 \times 10^{-11}$), often in combination with other motifs bound by Tpk1-modulated transcription factors, such as Adr1 (37/64; $p < 6 \times 10^{-3}$) and Cat8 (26/64;

$p < 2 \times 10^{-3}$). However, our method suggests that Tpk1 is an activator of the module in contrast to its known role as a repressor. We discuss this discrepancy in Section 7.3.

6.3.2 Global View

Our global evaluation is based on the same principles as the global analysis introduced in in Section 5.4. We employ an automated method to associate both modules and regulators with significantly enriched properties (hypergeometric p-value) in their respective gene sets. Modules are annotated based on their gene sets. Regulators are annotated based on their targets: since a regulator gene r may regulate more than one module, its targets consist of the union of the genes in all modules predicted to be regulated by r.

We evaluated all 50 modules to test whether the proteins encoded by genes in the same module had related functions. We scored the functional/biological coherence of each module by the percent of its genes covered by annotations significantly enriched in the module ($p < 0.01$). Most modules (31/50) contained a biologically coherent set of genes at a coherence level above 50% and only 4/50 had gene coherence below 30%. Note that the actual coherence levels may be considerably higher, as many genes are not annotated in current databases. Indeed, an in-depth inspection revealed many cases where genes known to be associated with the main process of the module were simply not annotated as such.

We next turned to the evaluation of the inferred regulation programs. We compared the known function of the inferred regulators with the method's predictions. In most modules (35/50), the regulators were predicted to play a role under the expected conditions. More importantly, most modules (30/50) also included genes known to be regulated by at least one of the module's predicted regulators. Furthermore, some modules (15/50) also had an exact match between *cis*-regulatory motifs enriched ($p < 10^{-4}$) in upstream regions of the module's genes and the regulator known to bind to that motif.

We obtained a global view of the modules and their function by compiling all gene annotations and motifs significantly enriched in each module into a single matrix [100] (see Figure 6.6 left). This presentation enables an automatic approach for deriving rich descriptions for modules. For example, the attributes for the respiration module (Figure 6.4) are immediately apparent in this representation, including the Hap4 and Msn4 (STRE) binding sites, and the ion transport, TCA cycle, mitochondrion, and aerobic respiration annotated genes (highlighted rectangles labeled 1). Note the enrichment of three annotations representing a biochemical process, cellular compartment, and physiological process, respectively - all relating to cellular respiration.

Analysis using the matrix representation can lead to many insights. For example, our algorithm resulted in four different amino acid (AA) metabolism modules. The submatrix for these AA metabolism modules (see Figure 6.7 (a)) supports their division into different modules: while the modules share certain attributes (e.g., a GCN4 motif characteristic of AA metabolism), each is characterized by a unique combination of gene annotations and cis-regulatory motifs (e.g., only the

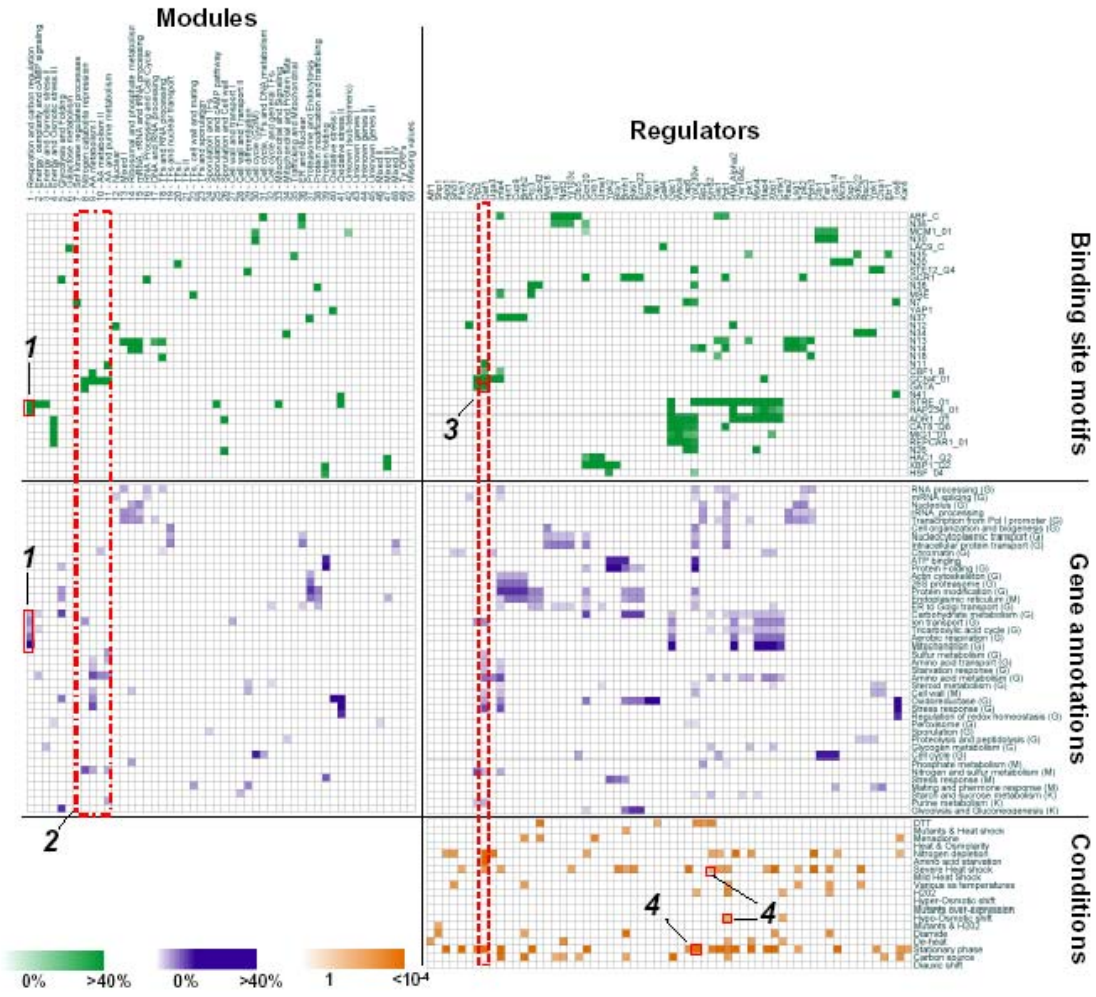

Figure 6.6: Global view of modules (left) and regulators (right). A matrix displaying modules verses their significanly enriched annotations/motifs. Only significantly enriched annotations ($p < 0.05$) are displayed. Binding sites (green) include both known motifs (TRANSFAC [48]) and de novo motifs (discovered by a motif-finding program [80]). Gene annotations (purple) were compiled from Gene Ontology (G) [16], MIPS (M) [69], and KEGG (K) [59]. The bottom right matrix displays significant conditions regulated by each regulator (brown).

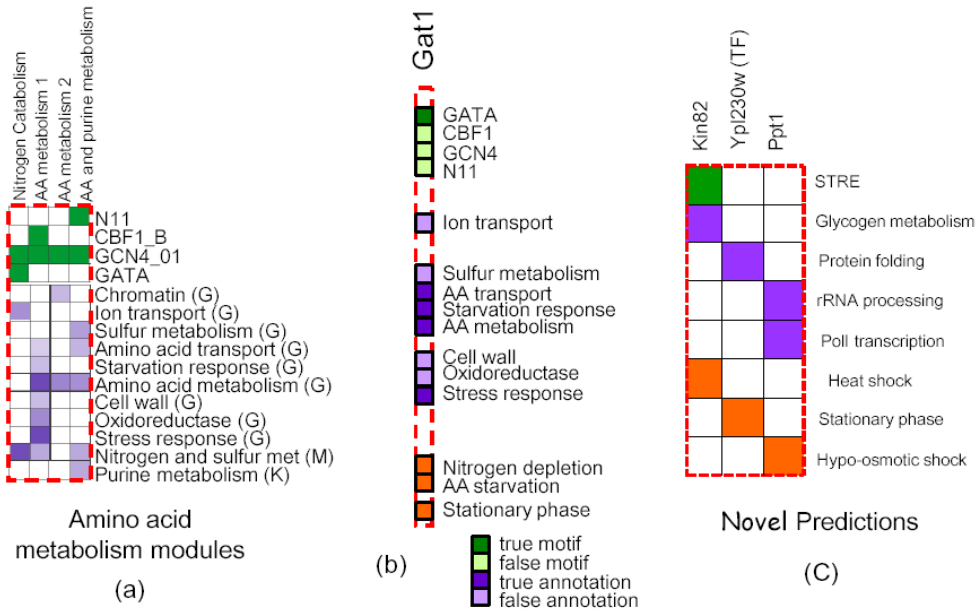

Figure 6.7: Highlights from the global matrix view: (a) submatrix of amino acid metabolism modules (Figure 6.6 highlighted rectangle labeled 2). (b) Annotation of Gat1 (Figure 6.6 highlighted rectangle 3). (c) Automated novel predictions (Figure 6.6 rectangles labeled 4).

Nitrogen catabolism module is also with the GATA motif characteristic of the nitrogen starvation response).

Similar to the global view for modules, we construct a global view for regulators associating each regulator with biological processes, possibly a binding motif and experimental conditions (see Figure 6.6 right). In addition to annotations and motifs, we can associate each regulator with the experimental conditions that it significantly regulates by examining how conditions are split by each relevant regulation tree. For example, in the Nitrogen catabolite module (Figure 6.5), Gat1 up-regulation distinguishes nitrogen depletion, AA starvation and stationary phase conditions from the rest (right branch), and would thus be associated with regulation in these conditions. For each occurrence of a regulator as a decision node in a regression tree, we computed the partition of each experimental condition between the right branch and the left branch, and used the binomial distribution to compute a p-value on this partition. The resulting associations with p-values < 0.05 are summarized in Figure 6.6 (bottom right matrix).

Each column in Figure 6.6 represents all the resulting associations corresponding to a regulator r. These result in a comprehensive but somewhat noisy annotation of that regulator. We take Gat1 as an example (see Figure 6.7 (b)). The matrix correctly suggests that Gat1 regulates nitrogen and sulfur metabolism processes, binds to the GATA motif, and works under conditions of nitrogen depletion. Alongside these correct associations are seemly incorrect associations (e.g. Gat1 binds

to GCN4 motif). These "noisy annotations" most likely result from the fact the each regulator is associated with *all* genes in each of the modules it regulates. Each such module is also associated with other regulators, thus we speculate that some of the noisy annotations might correspond to these other regulators. Contrary to motifs and annotations, experimental conditions are derived based on splits by the relevant regulator and are thus unique to that regulator.

6.3.3 Experimental Validation

When we consider uncharacterized regulators, the predicted regulator annotations provide focused hypotheses about the processes they regulate, the conditions under which they work, and the *cis*-regulatory motifs through which their regulation is mediated. For example, we can predict that the putative transcription factor Ypl230w regulates genes important for protein folding during stationary phase. The ability to generate detailed hypotheses, in the form "regulator r regulates process P under conditions W", is among the most powerful features of the module networks procedure, as it also suggests the specific experiments that can validate these hypotheses. Figure 6.7 (c) lists a number of such automatically generated predictions.

We selected three hypotheses suggested by the method, involving largely uncharacterized putative regulators. To test our ability to predict different types of regulatory mechanisms, we selected a putative zinc-finger transcription factor, Ypl230w, and two putative signaling molecules, the protein kinase Kin82 and the phosphatase Ppt1. Under normal growth conditions, all three deletion strains show no apparent abnormalities. As discussed above, each hypothesis generated by the method provides the significant conditions under which the regulator is active, and thereby specifies the experimental conditions under which the mutant should be tested (see Figure 6.7 (c)). Note, there are tens of different experimental conditions, therefore pin-pointing the conditions under which the regulator is active is crucial towards the success of the experiment.

These predictions were experimentally validated in collaboration with the Botstein lab at Stanford. They employed microarray analysis to compare the transcriptional response in the deletion strain to that of the wild-type strain, under the conditions predicted by our method. These genome-wide experiments enable a complete evaluation of the accuracy of our predictions for each regulator: whether it plays a regulatory role in the predicted conditions; whether it regulates genes in modules that it was predicted to regulate; and most importantly, whether it regulates processes that the method predicted it regulates.

For completion, we include the resulting analysis. A set of differentially expressed genes was derived from the microarray experiments for each of the tested regulators (see [82] for details). To test whether our method correctly predicted the targets of each regulator, we examined the distribution of the differentially expressed genes among the modules. For each putative regulator r, we calculated a p-value for the enrichment of differentially expressed genes in each module, and ranked the modules according to these p-values. In all three cases, the highest ranking module was predicted to be regulated by r (Figure 6.8 (a)). In each case, 25% (Ppt1, $p < 9 \times 10^{-3}$), 26% (Kin82, $p < 10^{-4}$), and 30% (Ypl230w, $p < 10^{-4}$) of the genes in the highest ranking module showed

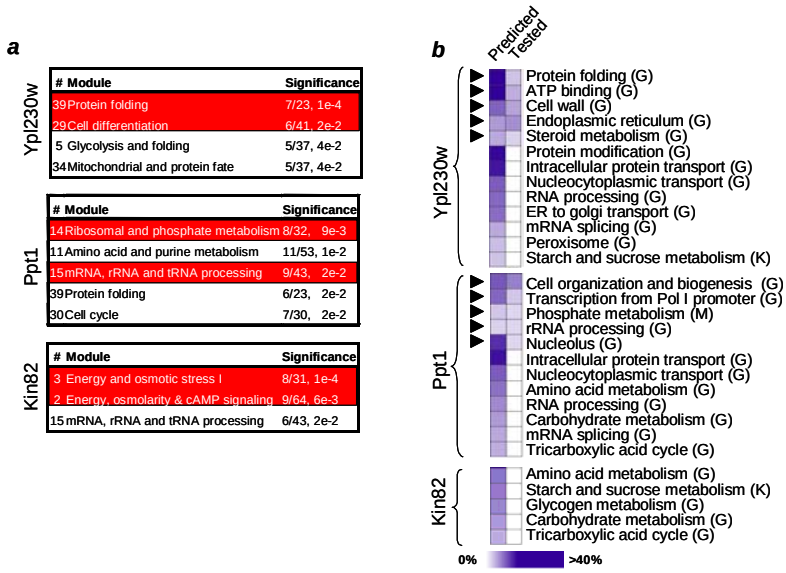

Figure 6.8: Validation of our method's predictions for putative regulators. (a) Ranked modules table for each tested regulator. Ranking is based on p-value calculated for enrichment of differentially expressed genes in each module Modules predicted to be regulated by the respective regulator are highlighted. (b) Functional predictions for tested regulators. The left column (Predicted) for each regulator shows all annotations predicted by the method to be associated with that regulator. The right column (Tested) shows which annotations were also significantly enriched in the set of differentially expressed genes of each regulator ($p < 0.05$; black triangles). The intensity of each entry represents the fraction of genes with the annotation from the set of differentially expressed genes.

differential expression.

Furthermore, we tried to identify the process regulated by each regulator, by searching for significantly enriched functional annotations in its set of differentially expressed genes. In two cases (Ypl230w, and Ppt1), the annotations matched those predicted for the regulator (Figure 6.8 (b)), supporting the method's suggestions for the regulatory roles of the tested regulators: Ypl230w activates protein folding, cell wall, and ATP binding genes, and Ppt1 represses phosphate metabolism and rRNA processing.

In summary, deletion of each of the three regulators caused a marked impairment in the expression of a significant fraction of their computationally predicted targets, supporting the method's predictions and providing important insight regarding the function of these uncharacterized regulators.

6.4 Definition and Scoring

Now that we have demonstrated that our module networks discovery method offers unique capabilities in extracting modularity and regulation from expression data. We provide a formal definition of module networks and technical details on how these can be learned from gene expression data.

6.4.1 Formal Definition

As described above, a module represents a set of variables that share the same set of parents and the same CPD. As a notation, we represent each module by a *formal variable* that we use as a placeholder for the variables in the module. A *module set* $\mathcal{C}$ is a set of such formal variables $\mathbf{M}_1, \ldots, \mathbf{M}_K$. In the gene expression domain, each $\mathbf{M}_i$ represents a regulatory module.

We represent by $Val(\mathbf{M}_j)$ the set of possible values of the formal variable of the j'th module. As all the variables in a module share the same CPD, they must have the same domain of values. In our case, all variables have the same domain. In our implementation, the expression of gene X is represented by two random variables, depending on the context. As a gene in a module, $Val(X)$ is continuous valued, representing the raw gene expression measurement. As a regulator, $Val(X)$ is one of three values: up-regulated, basel and down-regulated, representing the discretized value of the raw measurement.

A module network relative to $\mathcal{C}$ consists of two components. The first defines a template probabilistic model for each module in $\mathcal{C}$; all of the variables assigned to the module will share this probabilistic model.

Definition 6.4.1: A *module network template* $\mathcal{T} = (\mathcal{S}, \theta)$ for $\mathcal{C}$ defines, for each module $\mathbf{M}_j \in \mathcal{C}$:

- a set of parents $\mathbf{Pa}_{\mathbf{M}_j} \subset \mathcal{X}$;
- a *conditional probability template (CPT)* $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$ which specifies a distribution over $Val(\mathbf{M}_j)$ for each assignment in $Val(\mathbf{Pa}_{\mathbf{M}_j})$.

We use $\mathcal{S}$ to denote the dependency structure encoded by $\{\mathbf{Pa}_{\mathbf{M}_j} : \mathbf{M}_j \in \mathcal{C}\}$ and θ to denote the parameters required for the CPTs $\{P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j}) : \mathbf{M}_j \in \mathcal{C}\}$. ∎

In our example, we have three modules M_1, M_2, and M_3, with $\mathbf{Pa}_{\mathbf{M}_1} = \emptyset$, $\mathbf{Pa}_{\mathbf{M}_2} = \{\textit{Bmh1}\}$, and $\mathbf{Pa}_{\mathbf{M}_3} = \{\textit{Usv1}, \textit{Far1}\}$.

The second component is a module assignment function, that assigns each variable $X_i \in \mathcal{X}$ to one of the K modules, $\mathbf{M}_1, \ldots, \mathbf{M}_K$.

Definition 6.4.2: A *module assignment function* for $\mathcal{C}$ is a function $\mathcal{A} : \mathcal{X} \rightarrow \{1, \ldots, K\}$ such that $\mathcal{A}(X_i) = j$ only if $Val(X_i) = Val(\mathbf{M}_j)$. ∎

In our example, we have that $\mathcal{A}(\textit{Bmh1}) = 1$, $\mathcal{A}(\textit{Gic2}) = 2$, $\mathcal{A}(\textit{Usv1}) = 2$, and so on.

A module network defines a probabilistic model by using the formal random variables $\mathbf{M}_j$ and their associated CPDs as templates that encode the behavior of all of the variables assigned to that module. Specifically, we define the semantics of a module network by "unrolling" a Bayesian network where all of the variables assigned to module $\mathbf{M}_j$ share the parents and conditional probability template assigned to $\mathbf{M}_j$ in $\mathcal{T}$. For this unrolling process to produce a well-defined distribution, the resulting network must be acyclic. Acyclicity can be guaranteed by the following simple condition on the module network:

Definition 6.4.3: Let $\mathcal{M}$ be a triple $(\mathcal{C}, \mathcal{T}, \mathcal{A})$, where $\mathcal{C}$ is a module set, $\mathcal{T}$ is a module network template for $\mathcal{C}$, and $\mathcal{A}$ is a module assignment function for $\mathcal{C}$. $\mathcal{M}$ defines a directed *module graph* $\mathcal{G}_{\mathcal{M}}$ as follows:

- the nodes in $\mathcal{G}_{\mathcal{M}}$ correspond to the modules in $\mathcal{C}$;
- $\mathcal{G}_{\mathcal{M}}$ contains an edge $\mathbf{M}_j \rightarrow \mathbf{M}_k$ if and only if there is a variable $X \in \mathcal{X}$ so that $\mathcal{A}(X) = j$ and $X \in \mathbf{Pa}_{\mathbf{M}_k}$.

We say that $\mathcal{M}$ is a a *module network* if the module graph $\mathcal{G}_{\mathcal{M}}$ is acyclic. ∎

For example, for the module network of Figure 6.1(b), the module graph has the structure $\mathbf{M}_1 \rightarrow \mathbf{M}_2 \rightarrow \mathbf{M}_3$. We now define the semantics of a Module network.

Definition 6.4.4: A module network $\mathcal{M} = (\mathcal{C}, \mathcal{T}, \mathcal{A})$ defines a *ground Bayesian network* $\mathcal{B}_{\mathcal{M}}$ over $\mathcal{X}$ as follows: For each variable $X_i \in \mathcal{X}$, where $\mathcal{A}(X_i) = j$, we define the parents of X_i in $\mathcal{B}_{\mathcal{M}}$ to be $\mathbf{Pa}_{\mathbf{M}_j}$, and its conditional probability distribution to be $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$, as specified in $\mathcal{T}$. The distribution associated with $\mathcal{M}$ is the one represented by the Bayesian network $\mathcal{B}_{\mathcal{M}}$. ∎

Returning to our example, the Bayesian network of Figure 6.1(a) is the ground Bayesian network of the module network of Figure 6.1 (b).

Using the acyclicity of the module graph, we can now show that the semantics for a module network is well-defined.

Proposition 6.4.5: *If $\mathcal{G}_{\mathcal{M}}$ is a directed acyclic graph, then the dependency graph of $\mathcal{B}_{\mathcal{M}}$ is acyclic.*

Proof: The proof follows from the direct correspondence between edges in the module graph and edges in the ground Bayesian network. We construct a proof by contradiction. Assume that $\mathcal{G}_{\mathcal{M}}$ is acyclic, but $\mathcal{B}_{\mathcal{M}}$ is cyclic. Thus, there exists a path p of variables in $\mathcal{B}_{\mathcal{M}}$ of length $l \leq n$ edges such that $p_i \in p$ is a parent of $p_{i+1} \in p$ in $\mathcal{B}_{\mathcal{M}}$ for $i = 1, \ldots l-1$ and $p_1 = p_l$. We can convert p to a path in $\mathcal{G}_{\mathcal{M}}$ by replacing each node $p_i \in p$ with the module to which it belongs, $\mathbf{M}_{\mathcal{A}(p_i)}$. Our definition of a module graph guarantees that such a path exists in $\mathcal{G}_{\mathcal{M}}$ and thus it will be cyclic in $\mathcal{G}_{\mathcal{M}}$, contradicting our assumption that $\mathcal{G}_{\mathcal{M}}$ is acyclic. Hence, it follows that $\mathcal{B}_{\mathcal{M}}$ is acyclic. ∎

Corollary 6.4.6: *For any module network $\mathcal{M}$, $\mathcal{B}_{\mathcal{M}}$ defines a coherent probability distribution over $\mathcal{X}$.*

As we can see, a module network provide a succinct representation of the ground Bayesian network. A Bayesian network for the gene expression domain needs to represent thousands of CPDs. On the other hand, a module network can represent a good approximation of the domain using a model that uses only few dozen CPDs.

6.4.2 Bayesian Scoring

We now turn to the task of learning module networks from data. Recall that a module network is specified by a set of modules $\mathcal{C}$, an assignment function $\mathcal{A}$ of nodes to modules, the parent structure $\mathcal{S}$ specified in $\mathcal{T}$, and the parameters θ for the local probability distributions $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$. We assume in this paper that the set of modules $\mathcal{C}$ is given, and omit reference to it from now on. In this section we develop a scoring function, $\mathrm{score}(\mathcal{S}, \mathcal{A} : \mathcal{D})$, that measures how well each combination of structure and assignment function fits the observed data.

Our scoring function is based on the Bayesian paradigm and is developed similarly to the scoring function for Bayesian networks described in Section 2.3. We briefly follow this derivation highlighting the differences. In this section we treat multinomial CPTs in order to correspond more closely to Section 2.3. In Section 6.5.4 we describe how to adjust scoring and learning to the regression tree CPT used for the regulation programs.

6.4.3 Likelihood Function

We start by examining the *data likelihood* function

$$L(\mathcal{M} : \mathcal{D}) = P(\mathcal{D} \mid \mathcal{M}) = \prod_{m=1}^{M} P(\mathbf{x}[m] \mid \mathcal{T}, \mathcal{A}).$$

As the semantics of a module network is defined via the ground Bayesian network, we have that, in the case of complete data, the likelihood decomposes into a product of *local likelihood functions*, one for each variable. In our setting, however, we have the additional property that the variables in a module share the same local probabilistic model. Hence, we can aggregate these local likelihoods, obtaining a decomposition according to modules.

More precisely, let $\mathbf{X}^j = \{X \in \mathcal{X} \mid \mathcal{A}(X) = j\}$, and let $\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}}$ be the parameters associated with the CPT $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$. We can decompose the likelihood function as a product of *module likelihoods*, each of which can be calculated independently and depends only on the values of $\mathbf{X}^j$ and $\mathbf{Pa}_{\mathbf{M}_j}$, and on the parameters $\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}}$:

$$L(\mathcal{M} : \mathcal{D})$$

$$= \prod_{j=1}^{K} \left[\prod_{m=1}^{M} \prod_{X_i \in \mathbf{X}^j} P(x_i[m] \mid \mathbf{pa}_{\mathbf{M}_j}[m], \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}}) \right]$$
$$= \prod_{j=1}^{K} L_j(\mathbf{Pa}_{\mathbf{M}_j}, \mathbf{X}^j, \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D}) \quad (6.1)$$

The key difference is in calculating the sufficient statistics. In a module network, all of the variables in the same module share the same parameters. Thus, we pool all of the data from the variables in $\mathbf{X}^j$, and calculate our statistics based on this pooled data. More precisely, let $S_j(M_j, \mathbf{U})$ be a sufficient statistic function for the CPT $P(M_j \mid \mathbf{U})$. Then the value of the statistic on the data set $\mathcal{D}$ is

$$\hat{S}_j = \sum_{m=1}^{M} \sum_{X_i \in \mathbf{X}^j} S_j(x_i[m], \mathbf{pa}_{\mathbf{M}_j}[m]). \quad (6.2)$$

For example, in the case of multinomial table CPTs, we have one sufficient statistic function for each joint assignment $x \in Val(\mathbf{M}_j), \mathbf{u} \in Val(\mathbf{Pa}_{\mathbf{M}_j})$, which is $\eta\{X_i[m] = x, \mathbf{pa}_{\mathbf{M}_j}[m] = \mathbf{u}\}$ — the indicator function that takes the value 1 if the event $(X_i[m] = x, \mathbf{Pa}_{\mathbf{M}_j}[m] = \mathbf{u})$ holds, and 0 otherwise. The statistic on the data is

$$\hat{S}_j[x, \mathbf{u}] = \sum_{m=1}^{M} \sum_{X_i \in \mathbf{X}^j} \eta\{X_i[m] = x, \mathbf{Pa}_{\mathbf{M}_j}[m] = \mathbf{u}\}$$

Given these sufficient statistics, the formula for the module likelihood function is:

$$L_j(\mathbf{Pa}_{\mathbf{M}_j}, \mathbf{X}^j, \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D}) = \prod_{x, \mathbf{u} \in Val(\mathbf{M}_j, \mathbf{Pa}_{\mathbf{M}_j})} \theta_{x|\mathbf{u}}^{\hat{S}_j[x,\mathbf{u}]}.$$

This term is precisely the likelihood function presented in Eq. (2.3). The only difference is that the vector of sufficient statistics for a local likelihood term is pooled over all the variables in the corresponding module. For example, consider the likelihood function for the module network of Figure 6.1(b). In this network we have three modules. The first consists of a single variable and has no parent, and so the vector of statistics $\hat{S}[\mathbf{M}_1]$ is the same as the statistics of the same variable $\hat{S}[\textit{Bmh1}]$. The second module contains three variables, and we have that the sufficient statistics for the module CPT is the sum of the statistics we would collect in the ground Bayesian network of Figure 6.1(a): $\hat{S}[\mathbf{M}_2, \textit{Bmh1}] = \hat{S}[\textit{Usv1}, \textit{Bmh1}] + \hat{S}[\textit{Gic2}, \textit{Bmh1}] + \hat{S}[\textit{Far1}, \textit{Bmh1}]$. Finally, $\hat{S}[\mathbf{M}_3, \textit{Usv1}, \textit{Far1}] = \hat{S}[\textit{Hsp10}, \textit{Usv1}, \textit{Far1}] + \hat{S}[\textit{Hsp60}, \textit{Usv1}, \textit{Far1}]$.

As with Bayesian networks, the decomposition of the likelihood function allows us to perform maximum likelihood or MAP parameter estimation efficiently, optimizing the parameters for each module separately.

6.4.4 Priors and the Bayesian Score

In Section 2.3.1 we formulated a Bayesian score for each network structure $\mathcal{G}$. Here, in addition to structure, we also take the assignment function into account. Specifically, we define a model score for a pair $(\mathcal{S}, \mathcal{A})$ as the posterior probability of the pair, integrating out the possible choices for the parameters θ. We define an assignment prior $P(\mathcal{A})$, a structure prior $P(\mathcal{S} \mid \mathcal{A})$ and a parameter prior $P(\theta \mid \mathcal{S}, \mathcal{A})$. These describe our preferences over different networks *before* seeing the data.

We define the Bayesian score as the log of $P(\mathcal{S}, \mathcal{A} \mid \mathcal{D})$, ignoring the normalization constant

$$\begin{aligned} &\text{score}(\mathcal{S}, \mathcal{A} \ : \ \mathcal{D}) = \\ &\quad \log P(\mathcal{A}) + \log P(\mathcal{S} \mid \mathcal{A}) + \log P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) \end{aligned} \tag{6.3}$$

As with Bayesian networks, when the priors satisfy certain conditions, the Bayesian score decomposes. This decomposition allows to efficiently evaluate a large number of alternatives. The same general ideas carry over to module networks, but we also have to include assumptions that take the assignment function into account. Following is a list of conditions on the prior required for the decomposability of the Bayesian score:

Definition 6.4.7: Let $P(\mathcal{A})$, $P(\mathcal{S} \mid \mathcal{A})$, $P(\theta \mid \mathcal{S}, \mathcal{A})$ be assignment, structure, and parameter priors.

- $P(\theta \mid \mathcal{S}, \mathcal{A})$ satisfies *parameter independence* if

$$P(\theta \mid \mathcal{S}, \mathcal{A}) = \prod_{j=1}^{K} P(\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}, \mathcal{A}).$$

- $P(\theta \mid \mathcal{S}, \mathcal{A})$ satisfies *parameter modularity* if $P(\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}_1, \mathcal{A}) = P(\theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}_2, \mathcal{A})$ for all structures $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\mathbf{Pa}_{\mathbf{M}_j}^{\mathcal{S}_1} = \mathbf{Pa}_{\mathbf{M}_j}^{\mathcal{S}_2}$.
- $P(\theta, \mathcal{S} \mid \mathcal{A})$ satisfies *assignment independence* if $P(\theta \mid \mathcal{S}, \mathcal{A}) = P(\theta \mid \mathcal{S})$ and $P(\mathcal{S} \mid \mathcal{A}) = P(\mathcal{S})$.
- $P(\mathcal{S})$ satisfies *structure modularity* if $P(\mathcal{S}) \propto \prod_j \rho_j(\mathcal{S}_j)$ where $\mathcal{S}_j$ denotes the choice of parents for module $\mathbf{M}_j$, and ρ_j is a distribution over the possible parent sets for module $\mathbf{M}_j$.
- $P(\mathcal{A})$ satisfies *assignment modularity* if $P(\mathcal{A}) \propto \prod_j \alpha_j(\mathcal{A}_j)$, where $\mathcal{A}_j$ is the choice of variables assigned to module $\mathbf{M}_j$, and $\{\alpha_j : j = 1, \ldots, K\}$ is a family of functions from $2^{\mathcal{X}}$ to the positive reals. ∎

Parameter independence, parameter modularity, and structure modularity are the natural analogues of standard assumptions in Bayesian network learning (see Section 2.3.1). Two assumptions are new to module networks. Assignment independence makes the priors on the parents and parameters of a module independent of the exact set of variables assigned to the module. Assignment modularity implies that the prior on $\mathcal{A}$ is proportional to a product of local terms, one corresponding

to each module. Thus, the reassignment of one variable from one module $\mathbf{M}_i$ to another $\mathbf{M}_j$ does not change our preferences on the assignment of variables in modules other than i, j.

As for the standard conditions on Bayesian network priors, the conditions we define are not universally justified, and one can easily construct examples where we would want to relax them. However, they simplify many of the computations significantly, and are therefore very useful even if they are only a rough approximation. Moreover, the assumptions, although restrictive, still allow broad flexibility in our choice of priors. For example, we can encode preference (or restrictions) on the assignments of particular variables to specific modules. In addition, we can also encode preference for particular module sizes.

When the priors satisfy the assumptions of Definition 6.4.7, the Bayesian score decomposes into local *module scores*:

$$\text{score}(\mathcal{S}, \mathcal{A} \ : \ \mathcal{D}) = \sum_{j=1}^{K} \text{score}_{\mathbf{M}_j}(\mathbf{Pa}_{\mathbf{M}_j}, \mathcal{A}(\mathbf{X}^j) \ : \ \mathcal{D}) \tag{6.4}$$

Using priors that are *conjugate* to the parameter distributions often leads to closed form analytic formula of the value of the integral as a function of the sufficient statistics of $L_j(\mathbf{Pa}_{\mathbf{M}_j}, \mathbf{X}^j, \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D})$. For example, using Dirichlet priors leads to the formula given in Eq. (2.11), the only difference being that the sufficient statistics are collected over modules.

6.5 Learning Algorithm

Given a scoring function over networks, we now consider how to find a high scoring module network. This problem is a challenging one, as it involves searching over two combinatorial spaces simultaneously — the space of structures and the space of module assignments. We therefore simplify our task by using an iterative approach that repeats two steps: In one step, we optimize a dependency structure relative to our current assignment function, and in the other, we optimize an assignment function relative to our current dependency structure.

6.5.1 Structure Search Step

The first type of step in our iterative algorithm learns the structure $\mathcal{S}$, assuming that $\mathcal{A}$ is fixed. This step involves a search over the space of dependency structures, attempting to maximize the score defined in Eq. (6.3). This problem is analogous to the problem of structure learning in Bayesian networks. We use a standard heuristic search over the combinatorial space of dependency structures. We define a search space, where each state in the space is a legal parent structure, and a set of operators that take us from one state to another. We traverse this space looking for high scoring structures using a search algorithm such as greedy hill climbing.

An obvious choice of local search operators involves steps of adding or removing a variable X_i from a parent set $\mathbf{Pa}_{\mathbf{M}_j}$. (Note that edge reversal is not a well-defined operator for module

networks, as an edge from a variable to a module represents a one-to-many relation between the variable and all of the variables in the module.)

When an operator causes a parent X_i to be added to a module $\mathbf{M}_j$, we need to verify that the resulting module graph remains acyclic, relative to the current assignment $\mathcal{A}$. Note that this step is quite efficient, as cyclicity is tested on the module graph, which contains only K nodes, rather than on the dependency graph of the ground Bayesian network, which contains n nodes (usually $n \gg K$).

Also note that, as in Bayesian networks, the decomposition of the score provides considerable computational savings. When updating the dependency structure for a module $\mathbf{M}_j$, the module score for another module $\mathbf{M}_k$ does not change, nor do the changes in score induced by various operators applied to the dependency structure of $\mathbf{M}_k$. Hence, after applying an operator to $\mathbf{Pa}_{\mathbf{M}_j}$, we need only update the delta score for those operators that involve $\mathbf{M}_j$.

6.5.2 Module Assignment Search Step

The second type of step in our iteration learns an assignment function $\mathcal{A}$ from data. This type of step occurs in two places in our algorithm: once at the very beginning of the algorithm, in order to initialize the modules; and once at each iteration, given a module network structure $\mathcal{S}$ learned in the previous structure learning step.

Module Assignment as Clustering

In this step, our task is as follows: Given a fixed structure $\mathcal{S}$ we want to find:

$$\mathcal{A} = \mathrm{argmax}_{\mathcal{A}'} \mathrm{score}_{\mathbf{M}}(\mathcal{S}, \mathcal{A}' \ : \ \mathcal{D}) \tag{6.5}$$

Interestingly, we can view this task as a clustering problem. A module consists of a set of variables that have the same probabilistic model. Thus, for a given instance, two different variables in the same module define the same probabilistic model, and therefore should have similar behavior. We can therefore view the module assignment task as the task of clustering variables into sets, so that variables in the same set have a similar behavior across all instances.

However, there are several key differences between this task and a standard clustering task. First, in general, the probabilistic model associated with each cluster has structure, as defined by the CPT template associated with the cluster (module). Moreover, our setting places certain constraints on the clustering, so that the resulting assignment function will induce a legal (acyclic) module network.

Module Assignment Initialization

In the initialization phase, we exploit the clustering perspective directly, using a form of hierarchical agglomerative clustering that is tailored to our application. Our clustering algorithm can be thought

of as performing *model merging* (as in [31]) in a simple probabilistic model. In the initialization phase, we do not yet have a learned structure for the different modules. Thus, from a clustering perspective, we consider a simple naive Bayes model for each cluster, where the distributions over the different features within each cluster are independent and have a separate parameterization. We begin by forming a cluster for each variable, and then merge two clusters whose probabilistic models over the features (arrays) are similar. We continue to merge clusters until we construct a module network with the desired number of modules.

Module Reassignment

In the module reassignment step, the task is more complex. We now have a given structure $\mathcal{S}$, and wish to find $\mathcal{A} = \mathrm{argmax}_{\mathcal{A}'} \mathrm{score}_{\mathbf{M}}(\mathcal{S}, \mathcal{A}' : \mathcal{D})$. We thus wish to take each variable X_i, and select the assignment $\mathcal{A}(X_i)$ that provides the highest score.

At first glance, we might think that we can decompose the score across variables, allowing us to determine independently the optimal assignment $\mathcal{A}(X_i)$ for each variable X_i. Unfortunately, this is not the case. Most obviously, the assignments to different variables must be constrained so that the module graph remains acyclic. For example, if $X_1 \in \mathbf{Pa}_{\mathbf{M}_i}$ and $X_2 \in \mathbf{Pa}_{\mathbf{M}_j}$, we cannot simultaneously assign $\mathcal{A}(X_1) = j$ and $\mathcal{A}(X_2) = i$. More subtly, the Bayesian score for each module depends non-additively on the sufficient statistics of all the variables assigned to the module. (The log-likelihood function is additive in the sufficient statistics of the different variables, but the log marginal likelihood is not.) Thus, we can only compute the delta score for moving a variable from one module to another given a *fixed* assignment of the other variables to these two modules.

We therefore use a sequential update algorithm that reassigns the variables to modules one by one. The idea is simple. We start with an initial assignment function $\mathcal{A}^0$, and in a "round-robin" fashion iterate over all of the variables one at a time, and consider changing their module assignment. When considering a reassignment for a variable X_i, we keep the assignments of all other variables fixed and find the optimal legal (acyclic) assignment for X_i relative to the fixed assignment. We continue reassigning variables until no single reassignment can improve the score. An outline of this algorithm appears in Figure 6.9

The key to the correctness of this algorithm is its sequential nature: Each time a variable assignment changes, the assignment function as well as the associated sufficient statistics are updated before evaluating another variable. Thus, each change made to the assignment function leads to a legal assignment which improves the score. Our algorithm terminates when it can no longer improve the score. Hence, it converges to a local maximum, in the sense that no single assignment change can improve the score.

The computation of the score is the most expensive step in the sequential algorithm. Once again, the decomposition of the score plays a key role in reducing the complexity of this computation: When reassigning a variable X_i from one module $\mathbf{M}_{old}$ to another $\mathbf{M}_{new}$, only the local score of these modules changes. The module score of all other modules remains unchanged. The rescoring of

Input:
A data set $\mathcal{D}$
$\mathcal{A}_0$ *// Initial assignment function*
$\mathcal{S}$ *// Dependency structure*
Output:
$\mathcal{A}$ *// Improved assignment function*
Sequential-Update
$\mathcal{A} = \mathcal{A}_0$
Loop
For $i = 1$ to n
score$^* = \text{score}(\mathcal{S}, \mathcal{A} : \mathcal{D})$, $\mathcal{A}^* = \emptyset$
For $j = 1$ to k
$\mathcal{A}' = \mathcal{A}$ except that $\mathcal{A}'(X_i) = j$
If $\langle \mathcal{G}_\mathcal{M}, \mathcal{A}' \rangle$ is cyclic, **continue**
If $\text{score}(\mathcal{S}, \mathcal{A}' : \mathcal{D}) >$ *score**
$\mathcal{A}^* = \mathcal{A}'$, *score*$^* = \text{score}(\mathcal{S}, \mathcal{A}' : \mathcal{D})$
If $(\mathcal{A}^* \neq \emptyset)$
$\mathcal{A} = \mathcal{A}^*$
Until no reassignments to any of $X_1, \ldots X_n$
Return $(\mathcal{A})$
}

Figure 6.9: Outline of sequential algorithm for finding the module assignment function

Input:
$\mathcal{D}$ // *Dataset*
K // *Number of modules*
Output:
$\mathbf{M}$ // *A module network*
Learn-Module-Network
$\mathcal{A}_0$ = cluster $\mathcal{X}$ into K modules
$\mathcal{S}_0$ = empty structure
Loop $t = 1, 2, \ldots$ until convergence
$\mathcal{S}_t$ = Greedy-Structure-Search($\mathcal{A}_{t-1}, \mathcal{S}_{t-1}$)
$\mathcal{A}_t$ = Sequential-Update($\mathcal{A}_{t-1}, \mathcal{S}_t$);
Return $\mathbf{M} = (\mathcal{A}_t, \mathcal{S}_t)$

Figure 6.10: Outline of the *module network* learning algorithm. Greedy-Structure-Search successively applies operators that change the structure as long as each such operator results in a legal structure and improves the module network score

these two modules can be accomplished efficiently by subtracting X_i's statistics from the sufficient statistics of $\mathbf{M}_{old}$ and adding them to those of $\mathbf{M}_{new}$.

6.5.3 Algorithm Summary

To summarize, our algorithm starts with an initial assignment of variables to modules. In general, this initial assignment can come from anywhere, and may even be a random guess. We choose to construct it using the clustering-based idea described in the previous section. The algorithm then iteratively applies the two steps described above: learning the module dependency structures, and reassigning variables to modules. These two steps are repeated until convergence. An outline of the module network learning algorithm is shown in Figure 6.10. Each of these two steps — structure update and assignment update — is guaranteed to either improve the score or leave it unchanged. We can thus prove:

Theorem 6.5.1: The iterative module network learning algorithm converges to a local maximum of $\mathrm{score}(\mathcal{S}, \mathcal{A} : \mathcal{D})$.

6.5.4 Learning with Regression Trees

For the continous valued gene expression domain we use a conditional probability model represented as a *regression tree* [10]. A regression tree T for $P(X \mid \mathbf{U})$ is defined via a rooted binary tree, where each *node* in the tree is either a *leaf* or an *interior node*. Each interior node is labeled with a test $U = u$ on some variable $U \in \mathbf{Pa}_{\mathbf{M}_j}$ and $u \in Val(U)$. The parameters of T are the distributions associated with each leaf. In our implementation, each leaf ℓ is associated with a univariate Gaussian distribution over values of X, parameterized by a mean μ_ℓ and variance σ^2_ℓ.

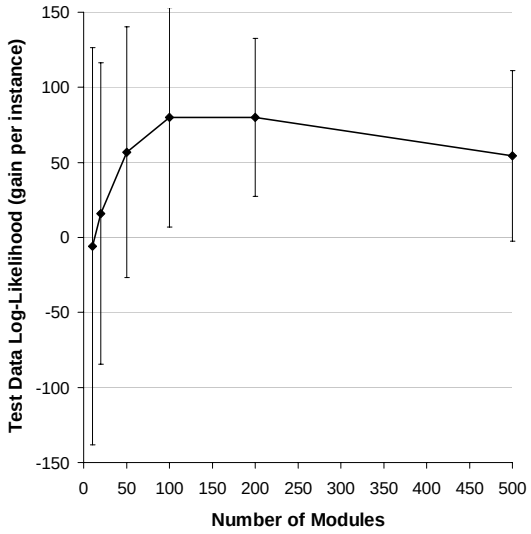

Figure 6.11: 10-fold cross validation comparing generalization ability between module networks of different sizes, using Bayesian networks as a baseline. For each run, zero is calibrated to be the Bayesian network performance and module network score is calculated relative to that base line. The error bars relate to the 10 different runs for each test dataset.

To learn module networks with regression-tree CPTs, we must extend our previous discussion by adding another component to $\mathcal{S}$ that represents the trees $T_1, \ldots, T_K$ associated with the different modules. Once we specify these components, the above discussion applies with several small differences. These issues are similar to those encountered when introducing decision trees to Bayesian networks [14, 33], and so we only briefly touch on them.

Given a regression tree T_j for $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$, the corresponding sufficient statistics are the statistics of the distributions at the leaves of the tree. For each leaf ℓ in the tree, and for each data instance $\mathbf{x}[m]$, we let $\ell_j[m]$ denote the leaf reached in the tree given the assignment to $\mathbf{Pa}_{\mathbf{M}_j}$ in $\mathbf{x}[m]$. The module likelihood decomposes as a product of terms, one for each leaf ℓ. Each term is the likelihood for the Gaussian distribution $\mathcal{N}(\mu_\ell; \sigma_\ell^2)$, with the sufficient statistics for a Gaussian distribution.

$$\begin{aligned} \hat{S}^0_{j,\ell} &= \sum_m \sum_{X_i \in \mathbf{X}^j} \eta\{\ell_j[m] = \ell\} \\ \hat{S}^1_{j,\ell} &= \sum_m \sum_{X_i \in \mathbf{X}^j} \eta\{\ell_j[m] = \ell\} x_i \\ \hat{S}^2_{j,\ell} &= \sum_m \sum_{X_i \in \mathbf{X}^j} \eta\{\ell_j[m] = \ell\} x_i^2 \end{aligned} \tag{6.6}$$

The local module score further decomposes into independent components, one for each leaf ℓ. Here, we use a Normal Gamma prior [25] for the distribution at each leaf: Letting $\tau_\ell = 1/\sigma_\ell^2$ stand for the precision at leaf ℓ, we define: $P(\mu_\ell, \tau_\ell) = P(\mu_\ell \mid \tau_\ell) P(\tau_\ell)$, where $P(\tau_\ell) \sim \Gamma(\alpha_0, \beta_0)$ and

$P(\mu_\ell \mid \tau_\ell) \sim \mathcal{N}\left(\mu_0; (\lambda_0 \tau_\ell)^{-1}\right)$, where we assume that all leaves are associated with the same prior. Letting $\hat{S}^i_{j,\ell}$ be defined as in Eq. (6.6), we have that the component of the log marginal likelihood associated with a leaf ℓ of module j is given by:

$$\begin{aligned} &-\tfrac{1}{2}\hat{S}^0_{j,\ell}\log(2\pi) + \tfrac{1}{2}\log\left(\tfrac{\lambda_0}{\lambda_0+\hat{S}^0_{j,\ell}}\right) \\ &+\log\left(\Gamma(\alpha_0 + \tfrac{1}{2}\hat{S}^0_{j,\ell})\right) - \log\left(\Gamma(\alpha_0)\right) \\ &+\alpha_0\log\left(\beta_0\right) - \left(\alpha_0 + \tfrac{1}{2}\hat{S}^0_{j,\ell}\right)\log\left(\beta\right) \end{aligned}$$

where

$$\beta = \beta_0 + \frac{1}{2}\left(\hat{S}^2_{j,\ell} - \frac{(\hat{S}^1_{j,\ell})^2}{\hat{S}^0_{\mathrm{J},\ell}}\right) + \frac{\hat{S}^0_{\mathrm{J},\ell}\lambda_0\left(\frac{\hat{S}^1_{\mathrm{J},\ell}}{\hat{S}^0_{\mathrm{J},\ell}} - \mu_0\right)^2}{2(\lambda_0 + \hat{S}^0_{\mathrm{J},\ell})}.$$

When performing structure search for module networks with regression-tree CPTs, in addition to choosing the parents of each module, we must also choose the associated tree structure. We use the search strategy proposed in [14], where the search operators are leaf splits. Such a *split* operator replaces a leaf in a tree T_j with an internal node with some test on a variable U. The two branches below the newly created internal node point to two new leaves, each with its associated Gaussian. This operator must check for acyclicity, as it implicitly adds U as a parent of $\mathbf{M}_j$. When performing the search, we consider splitting each possible leaf on each possible parent U and each value u.

6.6 Systematic Evalution

6.6.1 Cross Validation

We evaluated the generalization ability of different models, in terms of log-likelihood of test data, using 10-fold cross validation (see Section 5.6.1). In Figure 6.11, we evaluate the performance of module networks varying in the number of modules. As a baseline, we use the performance of Bayesian networks (no parameter sharing) for this data. To make the comparison valid, all networks, including Bayesian networks, are modeled using regression tree CPDs and are learned using the same candidate set of regulators and learning program. Thus, shared CPDs and the number of variables these were based on, was the only difference between models. Module networks generalize much better to unseen data for almost all numbers of modules, even for as many as 500 modules.

6.6.2 Gene Assignments

Our procedure initiates with a clustering of the genes based on their expression and then iterates between module assignment and regulation tree learning steps. For computational efficiency some iterations only optimize the parameters of the normal distributions at the leaves and leave the tree structure unchanged. An iteration that re-learns the regulation tree structure is employed only after

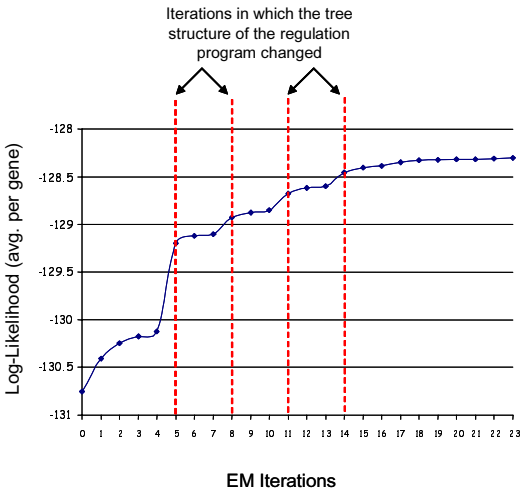

Figure 6.12: Likelihood of model at each iteration

iterations of module assignments and regulation tree parameter-optimizing converge. The results reported in this thesis converged after 23 iterations, 4 of which were tree structure change iterations.

Why not simply learn our regulation programs on the initial clusters? Do we gain anything by re-assigning genes to new modules and iterating the learning? In this section we will show that our iterative procedure is superior to simply learning regulation programs on the results of standard clustering. In Figure 6.12 demonstrates how the score of the resulting model improves in each iteration. As expected, the biggest jumps in the score are after tree structure learning.

How similar are the resulting modules (i.e. their gene composition) to the initial clusters? Overall, 49% of the genes changed their assignment as compared to the initial clusters. Figure 6.13 plots the percent of genes that changed in each iteration compared to the previous one and compared to the initial clusters. Note, the first iteration, over 35% of the genes change their module assignment.

The resulting modules are more biologically coherent than the initial clusters. We computed the enrichment of each GO annotation in the initial clusters and in the resulting modules. In Figure 6.14 we plotted the negative log of the best p-value found for each annotation in the resulting modules, against the negative log of the best p-value for that annotation in the initial clusters. This plot shows a clear shift in the biological coherence in favor of the resulting modules.

The iterations of our algorithm play a similar role to the iterations in the psi-blast [] algorithm. Just as psi-blast converges to protein motifs of a protein family, our procedure converges to *module signatures* of regulatory modules. Recall the Nitrogen Catabolite Repression Module, which has tight co-expression only when Gat1 is over-expressed. While in standard clustering each sample has equal weight, our scoring method re-weights the samples so that the Gat1 overexpressed samples determine most of the score. Thus, condition specific modules are formed.

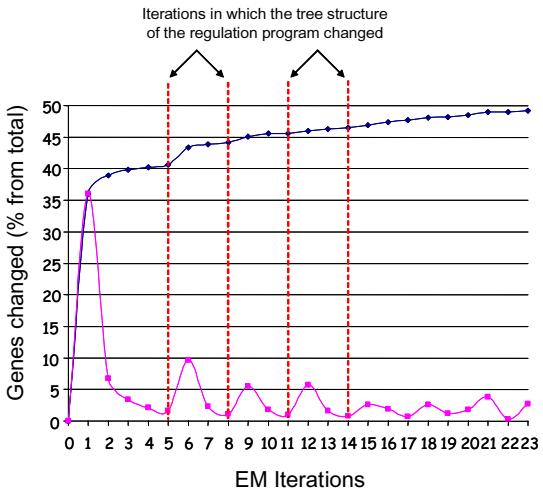

Figure 6.13: Percent of genes changing module assignments at each iteration

6.7 Discussion

Discovering biological organization from gene expression data is a promising but challenging task. The contribution in this chapter is two-fold. The module networks discovery method presented here offers unique capabilities in extracting modularity and regulation from expression data. Furthermore, the framework of *module networks* developed for this task can be successfully applied to other domains (e.g. stock data [81]).

The statistical advantages of module networks allow us to learn detailed regulatory programs over thousands of gene solely from approximately one hundred gene expression data samples. Overall, our the resulting model provides a clear global view of functional modules and their regulation, that corresponds well to the known biological literature. Perhaps the most powerful feature of our method is its ability to generate detailed testable hypotheses concerning the role of specific regulators and the conditions under which this regulation takes place. Microarray experiments planned based on the computational predictions for three putative regulators with as yet unknown functions (a transcription factor and two signaling molecules) validated the method's predictions and provided important insight regarding the function of these uncharacterized regulators. As more diverse gene expression datasets become available, it is our belief that applying the module networks may result in important new insights in the ongoing endeavor to understand the complex web of biological regulation.

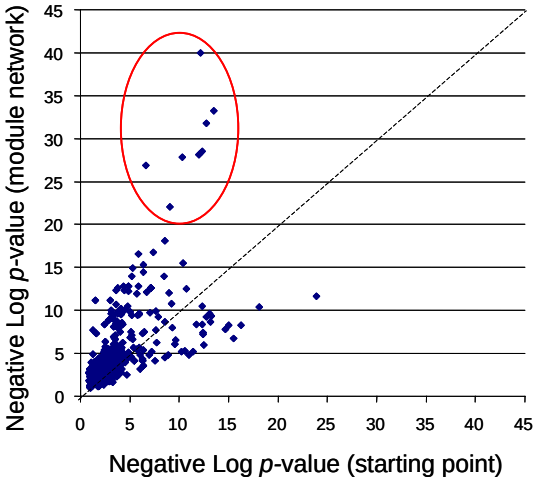

Figure 6.14: Biological coherence of modules verses initial clustering. waiting from fig from Eran

Chapter 7

Discussion

7.1 Summary

In this dissertation, we have seen three different methods to infer cellular pathways from gene expression profiles. The first approach uses ensembles of Bayesian networks to detect strong statistical dependencies between genes and based on these dependencies, reconstructs detailed subnetworks of interacting genes. The other two approaches (MinReg and Module Networks) focus the inference on regulatory relations by constraining the parents in the network to a pre-defined candidate set of known or putative regulators. The MinReg algorithm constructs a global regulation model by identifying a small set of "active" regulators and the genes that they regulate. The gene expression of this small set of chosen regulators can predict the behavior of the majority of the genes in the data. The Module Network approach is based on revealing coordinated co-regulation by identifying "regulatory modules": sets of co-regulated genes and the regulatory program that controls them.

No less important than the model itself and algorithms to learn the model from the data, is how to interpret and validate of the resulting model in a biological context. For each of the three methods, we attempted to answer the following questions.

- What is the biological interpretation of the features in the model?
- How well does the model correspond to existing biological knowledge?
- What novel biological insights do we gain from the inferred model?

We validated the reconstructed models by comparing them to independent data sources. We evaluated the resemblance between the interactions inferred by our methods to known biological interactions. These included interactions automatically extracted from the YPD [21] database and manually extracted from review papers. Since only a small fraction of the true biological relations are known, such a comparison is sparse in its scope, Therefore, we used gene annotations for both biological process and molecular function obtained from the SGD [11] and YPD [21] databases, to help support the biological plausibility of the inferred model. This analysis was further corroborated

using evidence from published journals. In addition, independent data sources provided further affirmation for our models. These include known *cis*-regulatory motifs [48] in the promoter regions, protein-protein interactions from DIP [101] and DNA-protein interactions [62].

For each method, our systematic evaluation confirms many aspects of the reconstructed model. The Subnetwork approach identified a number of subnetworks that correspond to coherent biological processes, including the mating pathway, low osmolarity cell wall pathway and amino acid metabolism (Chapter 3). The MinReg algorithm automatically associated both transcription factors and signaling molecules with their known targets and with the biological processes they are known to regulate (Chapter 5). The Module Network approach discovers known functional modules and the genes known to regulate them. The validation of our methods culminated in experimental confirmation of three automatically generated hypotheses. The Module networks allows to make *testable hypotheses* of the form "protein X regulates a module of genes G, under conditions C". Gene expression profiles of knockout strains in the specific conditions where the regulator is predicted to be active, confirmed the computational predictions for three putative regulators with as yet unknown functions (a transcription factor and two signaling molecules), providing important insight regarding the function of these uncharacterized regulators. Overall, our results demonstrate that it is possible to infer metabolic pathways, regulatory relationships, signaling cascades and an organization into co-regulated modules using gene expression data as input.

7.2 Comparing the Methods

The main differences between the methods involve the various strategies by which they tackle the issue of statistical robustness and the trade-offs these impose. One of the greatest challenges of the reconstruction of molecular pathways from gene expression data is the statistical significance of the resulting model. The domain is high dimensional (even simple organisms such as yeast have thousands of genes) and it contains a vast number of hidden and unobserved variables; yet we attempt to learn it using only a small number (hundreds) of noisy samples. Thus we must choose different tradeoffs, such as few interactions at fine detail verses many interactions at a coarser level or accuracy of the regulatory interactions themselves verses accuracy of the regulatory program. Some of these differences are summarized in Table 7.1.

In the subnetwork approach, we do not have confidence in the entire network, instead, we extract the common features on which many high scoring structures agree. This results in a fine detailed network structure, but only for a small set of interactions. In order to build a global network over all genes, the regulatory methods (MinReg and Module networks) reconstruct at a coarser, more abstract scale. These reconstructions are not always correct at the resolution of a single interaction, rather, they are more reliable when considering properties (e.g. annotations) of entire sets of co-regulated genes. Both regulatory methods gain part of their statistical robustness by limiting the total number of regulators (parents) in the model. MinReg, explicitly, by definition, and Module networks, implicitly. This forces the learning to identify the most pronounced regulators in the

	Bayesian Networks	MinReg Model	Module Networks
Model Scope	Subnetworks	Global	Global
Prior Assumptions	No	Yes	Yes
Robustness Strategy	Bootstrap	Few regulators	Common regulatory programs
Resolution of Structure	Finest	In-between	Coarse
Logic of Regulation	Weak	Exists	Strong

Table 7.1: Table comparing Bayesian networks, MinReg and Module networks

dataset. Regulators which effect only a small number of targets are not detected by these methods. Module networks gain additional robustness when learning the combinatorial logic of the regulatory programs by using parameter sharing and by pooling data samples of co-regulated genes together.

Our adaptation of Bayesian networks to gene expression does not rely on knowledge or assumptions specific to the biological domain. Our solutions are general: the sparse candidate algorithm (Section 4.1) dealt with the vast number of variables, non-parametric bootstrap (Section 3.3) dealt with the small number of samples and we treated mutations as ideal interventions (Section 4.2). Since apart from bias for small indegree, the Bayesian networks were not constrained to any particular preconception, these captured aspects of cellular pathways that were completely missed in the regulatory methods. Specifically, Bayesian networks reconstructed a number of surprisingly accurate metabolic pathways, often linking genes whom are one metabolic step from each other (Figure 3.9 and Section 3.7.2). This reconstruction of metabolic links raises many questions concerning the regulation of metabolism, such as understanding the mechanism behind the significant correlation between distance in metabolic pathways and coexpression.

The fundamental concept behind both MinReg and Module networks is quite similar. Both reconstruct global models of gene regulation and both focus the inference on transcriptional regulation by constraining the parents to a candidate list of known and putative regulators. They further narrow the search space, by using biologically motivated assumptions to constrain the structure of the learned networks. MinReg is constrained to find a model with a small number of “active” regulators. The Module networks approach makes the additional assumption that the genes are grouped into modules that share a common regulatory program. These assumptions lead to an abstraction of the network that misses the more intricate wiring of the network.

This “disadvantage” can also be viewed as an advantage, as it captures a higher level organization of the network. Even if it were feasible to reconstruct an exact network of interactions from the given data, this detailed wiring diagram over thousands of genes would be almost as hard to understand as the raw gene expression data itself. A higher order organization is often more easy to interpret: it helps elucidate not only which “wires” exist, but also what they do. In the analysis of the MinReg model, we do not focus on the single regulator target relationships, rather, we associate each regulator with Gene Ontology terms significantly enriched in its “regulatee set”. This approach highlights which biological processes and functions a particular gene is regulating. The Module networks procedure explicitly learns this higher order organization of network structure

by automatically organizing genes into co-regulated modules, each responsible for a common response or process, By doing so explicitly, the resulting modules have more coherent annotations (e.g. percentage of module genes that belong to a dominant annotation), than MinReg's regulatee sets.

Another strength of the Module network approach is due to the parameter sharing between genes in the same module, greatly enhancing the statistical robustness of the inferred model. Requiring similar behavior for genes in the same module is the main reason that the resulting gene sets are more coherently annotated than the MinReg regulatee sets. Furthermore, enforcing a common, condition specific regulatory program leads to superior coherence than in standard clustering. Most importantly, pooling many samples together enables to learn not only the structure of the regulatory network, but the regulatory program itself. Instead of inferring regulatory functions based on only 100 samples, the regulatory functions are reconstructed based on 3000 samples, a significant difference. Thus, the resulting regulatory programs are richer and include combinatorial logic between regulators. For example, the Respiration module is activated in conditions of Hap4 upregulation *OR* Msn4 upregulation. This combinatorial OR is further supported by the existence of binding sites to both Hap4 and Msn4 in many of the module's genes.

While the assumption of a shared regulatory program has many advantaged, it comes at a cost. In Module networks, each gene belongs to only one module and all genes in a module share the exact same regulatory program under all conditions. This is a over-simplification of biological modules, that are overlapping and dynamic in nature. Under different conditions, different sets of genes are coordinated to act together. Genes that function in a common process often have overlapping, but distinct regulator sets. In this respect, the MinReg approach has an advantage, as it assigns each gene a unique set of regulators chosen from a common pool. This is most clearly demonstrated by the Nitrogen catabolism response, that was reconstructed by both models. MinReg correctly pieced together the joint regulation of nitrogen starvation, it detected the GATA transcription factors: Gat1, Uga3 and Dal80, all whom are known to participate in the regulation of this response (see Figure 5.4). However, the molecular functions they regulate only partially overlapped, highlighting their specific roles in a common response. The Module network procedure only inferred Gat1 as a regulator of this response, Uga3 and Dal80 are suggested as members, rather than regulators, of the module (see Figure 6.5). In Figure 6.5, it is clear that the genes are only tightly co-regulated under Gat1 upregulation. In other conditions, the genes seem to have a diverse set of behaviors and regulatory programs.

7.3 From Gene Expression to Transcriptional Regulation

A critical challenge for the validity of our approach lies in explaining how regulatory events can be reconstructed from gene expression data. In this section we discuss our hypothesis as to how this is possible.

Our methods detect regulatory relations based on the statistical associations in gene expression.

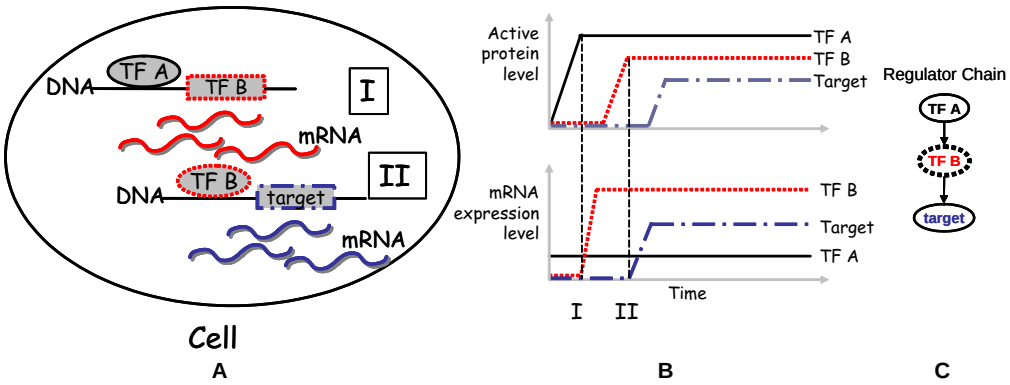

Figure 7.1: Regulator chain $A \hookrightarrow B \hookrightarrow C$(A) Cartoon of the cell at two time points. At time I, transcription factor (TF) A activates transcription factor B, raising the levels of mRNA and protein for transcription factor B. In time II, transcription factor B activates its target. (B) Time series plotting the protein and mRNA levels. (C) Network motif. The gene with the dotted border is predicted as regulator.

In order to capture a regulation event in gene expression data, we must observe concordant changes in the expression of both the regulator and its targets. Therefore, our approach relies on the assumption that the gene expression profile of the regulators provides evidence as to their activity level. This assumption is currently part of an ongoing biological debate, many researchers do not believe gene expression data can reveal the actual regulators themselves. Indeed, due to the complex, often post-transcriptional nature of regulation, our assumption does not always hold. To the contrary, recent large-scale analysis of the regulatory networks of *E. Coli* [87] and *S. Cerevisiae* [62] revealed the prevalence of cases in which the regulators are themselves transcriptionally regulated, a process whose functional importance is supported both theoretically and experimentally [67]. Such concordant changes in the expression of both the regulator and its targets might allow our automated procedure to detect statistical associations between them.

When a transcription factor is transcriptionally regulated, its mRNA expression might correlates well with its activity. Therefore, in cases where transcription factor A activates transcription factor B, which in turn activates some target C, our approach can possibly capture the regulatory relationship between B and C (see Figure 7.1). Note that steady state gene expression profiles do not observe temporal changes (as those plotted in the Figure 7.1). Instead, perturbations of the cell state (mutations and environmental treatments) create statistical associations between regulator and target, these are detected by our learning algorithm.

Figure 7.1 provides a very simplistic view of gene regulation. In many cases, gene expression data provides only part of the story: while transcription factors directly control transcription, the factors themselves are frequently regulated at the post-translational level. Furthermore, many transcription factors are active at very low levels of expression, and thus can not be detected reliably with microarrays. In such cases, even if the transcription factor is regulated transcriptionally, current

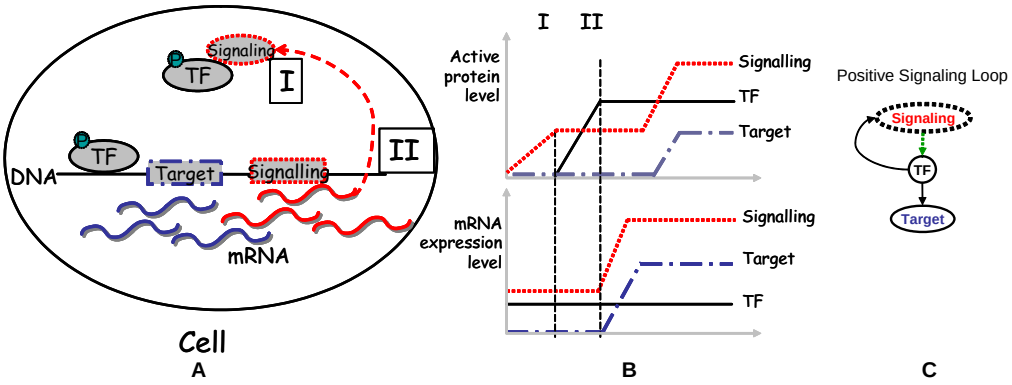

Figure 7.2: Signaling, positive feedback loop (A) In time I, the signaling molecule activates the transcription factor (TF). In time II, the transcription factor activates all its targets including the signaling gene. Therefore, the expression of the signaling molecule correlates with its indirect targets (B) Time series plotting the protein and mRNA levels. (C) Network motif

microarray technology can not observe its change. In these cases, our basic assumption does not hold. While the gene expression data is inherently oblivious to regulation of this kind, an indirect regulatory relationship can be detected. Sometimes our method identifies the regulatory relationship between a signal transduction molecule and its indirect targets.

We attribute this ability to the presence of positive and negative feedback loops: signaling molecule post-transcriptionally activates transcription factor, which in turn activates its targets. If the signalling molecule is target of the transcription factor, a feedback loop is formed (see Figure 7.2). In the first step, the cell contains an in-active form of the transcription factor and its targets are not transcribed. Small amounts of the signalling molecule activate the transcription factor without inducing any change to it's mRNA level. The actived transcription factor induces the transcription of all its targets, including the signalling molecule. While there is no detectable change in the expression of the transcription factor, the expression of the signaling molecule is concordant with its indirect targets.

Shen-Orr et.al. [87] use known *E. coli* regulatory relations to break down the design of the *E. Coli* transcriptional network into basic building blocks. They define *network motifs* as patterns of interconnections that recur in the transcriptional network at frequencies much higher than those found in randomized networks. They find that a number of such sub-structures include regulators who are themselves transcriptionally regulated. Lee et.al. [62] construct a genome wide regulatory map for *S. Cerevisiae* using a new high throughput experimental approach: *cis*-regulatory location analysis. This technology measures the binding of transcription factors to the promoter regions of an entire genome. Using genome-wide location data for 106 *S. Cerevisiae* transcription factors they found prevalence of many of the motifs previously detected in the *E. coli* regulatory network.

We used the data of Lee et.al. [62] to test if some of our inferred regulator-regulatee relationships

participate in such *network motifs*[1]. Indeed, many of the inferred regulatory relations are part of such regulatory motifs. Note, while the location data was obtained under normal growth conditions, the gene expression data was obtained under stress conditions and mutated strains (that are often under stress). Since regulation under stress conditions substantially differs from normal growth, many of the regulatory events in the gene expression dataset are missed by the *cis*-binding dataset.

We present different types of regulatory components and an example in which such a component is found (these are summarized in Figure 7.3):

- **(a)Regulator chain:** In this chain, a primary transcription factor activates a secondary transcription factor by enhancing its transcription. After the secondary transcription factor is translated into protein, it activates its own targets. In steady state expression profiles, this can result in a statistical association between the secondary transcription factor and targets of both the primary and secondary transcription factors. Note, in regulator chains, only the secondary transcription factor is inferred as a regulator. For example, the transcription factor, Phd1, activates a secondary transcription factor, Hap4. The module networks procedure found twenty one genes[2] that are bound by Hap4 in the location dataset, to be part of the respiration module which Hap4 regulates. Note, Pet9 was also inferred to be regulated by Hap4, while based on the location data it is regulated by Phd1. In this case our method possibly detected co-regulation instead of regulation. Such cases will be further discussed below.
- **(b) Auto-regulation:** A transcription factor activates both its own transcription and that of its target genes, thus, the transcription factor is co-expressed along with its targets. For example, Yap6 activates its own transcription and that of its target genes[3]. These were also inferred as part of the Snf kinase regulated processes module which Yap6 regulates.
- **(c) Positive signaling loop:** A signaling molecule activates (post transcriptionally) a transcription factor which induces the transcription of various targets, possibly including the signaling molecule. The coordinated expression changes in signaling molecule and its indirect targets allow the signaling molecule (but not the transcription factor) to be correctly inferred as a regulator. For example, the MAP kinase Slt2 activates Rlm1. The MinReg algorithm inferred Slt2 to regulate both Rlm1 and a number of its known targets: (Pst1, Crh1, Bop1, Ktr2, Gsc2, Yps3, Prp2). Evidence [96], including a Rml1 binding site in the Slt2 promotor suggests that Slt2 itself is also activated by Rlm1.
- **(d) Negative signaling loop:** Similar to the previous example, a negative feedback loop is also possible: a signaling molecule inhibits activity of a transcription factor, which induces transcription of its targets and possibly of signaling molecule. However, since both signaling molecule and the targets are up-regulated, the method predicts that the signaling molecule's

[1] Our comparison was made primarily to the Module networks method reconstruction, as this data did not exist during the course of the previous projects.

[2] Cox4, Cox6, Atp17, Cox7, Cox8, Qcr2, Mir1, Qcr7, Cox12, Qcr9, Cox13, Cyt1, Atp1, Atp2, Atp3, Rip1, Atp5, Atp7, Atp20, Cor1, and Ylr294c

[3] Hxt12, Hxt15, Hxt16, Yil122w, Fsp2, Yol157c, Yil172c, and Kel2

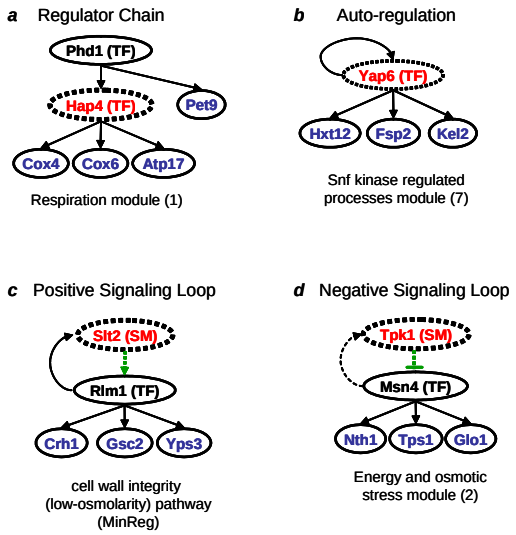

Figure 7.3: Network motifs: For each regulatory component, the relevant transcription factors (TF), signal transduction molecules (SM), target genes and their relations are shown. Cis-regulation events via transcription factors are shown in solid arrows, post-transcriptional events via signaling molecules in dotted arrows. The gene with dotted border is that predicted as a regulator by our method. The solid bordered regulators are not expected to be inferred from gene expression data. The targets included are those genes that are both predicted by our method to be regulated by the dotted regulator and also bound by the transcription factor according to the *cis*-location data ($p < 0.001$ in Lee et al. [62]).

> role is activation, in contrast to its actual inhibitory role. For example, Tpk1 inhibits the activity of Msn4. Msn4 regulates the targets: Nth1, Tps144, and Glo145. These are part of the Energy and osmotic stress module regulated by Tpk1. Tpk1's upstream region includes the Msn4 bound STRE motif, supporting Tkp1's regulation by Msn4. Indeed, Tpk1 is inferred as an activator rather than a repressor

Overall, our results demonstrate that regulatory events, including post-transcriptional ones, have a detectable signature in the expression of genes encoding transcription factors and signal transduction molecules. Nevertheless, the ability of our methods to discover regulatory events is due in part to secondary effects. The methods do not directly detect the post-translational activation of a transcription factor, rather, due to a feedback loop, they identify the resulting change in the transcription of the signaling molecule. Similarly to feedback signaling loops, we believe that our detection of direct metabolic links (see Section 3.7.2) may be due to feedback loops involving the metabolites themselves. This is supported by the discovery of such a feedback loop in the Galactose pathway [53].

Despite the successes described above, our methods fails to identify regulatory relations. These false negatives are divided between cases when the regulator's expression does not change sufficiently to merit detection, and cases where our method faile to identify the correct regulatory

relation, despite a detectable change in the regulator's expression pattern. A close examination of the relationships missed by our method reveals four categories of false negatives. The first two are relevant to all three reconstruction methods.

- **Undetectable regulators:** If the change in a regulator's activity is attributed exclusively (or mostly) to post-transcriptional changes, we do not expect our method to capture it. For example, the Dal81 gene is a known transcriptional activator of allantoin, GABA, and urea catabolic genes, when nitrogen sources are low (e.g., in later stationary phase). Although such conditions are included in our data set, there is little change in the expression of the Dal81 gene itself. Thus, our methods could not detect Dal81-mediated regulation. Furthermore, if the regulatory event and the concomitant change in the regulator's expression occur primarily under very specific conditions, that are not included in our data set, we do not expect our methods to capture it. While missed in our current analysis, these relations can be captured by our methods given a more appropriate dataset.
- **Regulator redundancy:** If several regulators participate in the same regulatory event, due to "explaining away" we expect our methods to capture only some representatives, missing the remaining regulators. This limitation applies both when several transcription factors work in one complex and when several signal transduction molecules and/or transcription factors are concatenated in one regulator chain. For example, the Hap2/3/4/5 transcription factors work in a ternary complex to regulate the expression of respiration genes. While the module networks procedure correctly captured Hap4 as a regulator of respiration it failed to identify Hap2, 3, and 5 that have a "redundant function". Note that the changes in Hap4's expression are the most pronounced among these four potential regulators, explaining the method's specific choice. Note that this limitation of our methods does not apply to combinatorial regulation, when several regulators have only partly overlapping roles (e.g., Hap4 and Msn4 in the respiration module).

The next two are specific to the two regulatory methods

- **Co-regulation redundancy:** Many modules contain target genes that also happen to belong to the candidate regulator set. As these genes often have an expression profile which is similar to that of the other target genes in the module, they may mistakenly be chosen as a regulator for the module, in some cases rendering the true regulator redundant. In such cases, the true regulator is often assigned as a module member, along with its targets. For instance, the MinReg algorithm inferred Apg1, a signaling molecule involved in induction of autophagy, as regulator of many Tor1 targets. Indeed, Apg1 is known to be regulated by Tor1 [78]. Tor1 itself is regulated post-transcriptionally, thus, we are captured Apg1 as its "replacement" in our model, reflecting co-regulation rather than true regulation.
- **Gene specific regulation:** Even when a regulator's expression pattern is highly (and possibly uniquely) predictive of that of its known target, this relationship may be specific to a single

target, and cannot be generalized to an entire module or set of genes. In such cases, neither MinReg nor Module networks, which are aimed to identify shared regulatory responses, would detect the regulatory relation. We believe that the statistical robustness and biological conciseness gained by taking a general perspective of regulation outweighs this particular limitation of our approach.

Thus, while we attempt to limit the false positives, our reconstruction approach inherently leads to many false negatives. We do not reconstruct the entire regulatory program of an entire organism, rather only certain aspects of it. Further work is required in order to explain why a specific transcription factor is chosen over other potential ones, why a signaling molecules is identified as a regulator instead of its cognate transcription factor in certain cases but not in others (e.g. Tpk1 and Msn4, both of which have a strong expression signature), or why a particular combination of regulators has been selected. While some of the reasons may have to do with our computational method, others may be related to critical biological issues such as the strength of feedback regulation and the coordination of action between various regulators. Answers to these questions can both aid the design of better regulatory models and algorithms to reconstruct them and can illuminate the regulation of regulators and the coordination of their actions.

7.4 Future Prospects

The task of cellular pathway reconstruction is one of the biggest challenges of the post genomic era. The work described in this dissertation is only a one step towards the goal of automated reconstruction of cellular pathways. Many specific issues and future direction have already been raised in the summary of each chapter, here we provide a more general overview.

One suggestion for the next step in our quest for regulatory modules is “dynamic module networks”. One way to implement this is a hybrid between MinReg and Module networks: a method that identifies a minimal set of common regulatory rules. Each such rule would be shared by many genes and its parameters would be estimated using parameter sharing. But, each gene would be regulated by a its own specific set of rules, giving it a unique signature of biological processes and responses.

We focused on the structure and connections within cellular pathways. We used standard distributions such as multinomials and regression trees for the local probability model that describes how a set of regulators affect their targets. Bayesian network analysis for genetic networks has already been enhanced by the use of non-parametric local CPDs [54] and more sophisticated models based on the chemistry of molecular interactions are being developed [70]. Many of these models are based on time series data [90, 58], that is beyond the scope of this thesis. Clearly time can help elucidate causality of regulation, yet there is very little work that takes the temporal aspect of the data into consideration [72, 5].

The methods described in this thesis use only gene expression as input, and are thus capable of reconstructing certain aspects of the cell’s activity. Today, in addition to DNA microarrays,

many new technologies can probe different attributes of the cell at a genome-wide scale and more technologies are in constant development. Mass spectrometry [56] and protein arrays [65] quantitatively measure abundance of proteins and other molecules in the cell. CIS-regulatory regions can be studied using motif analysis [94, 6] and location binding assays [88, 62]. Protein interactions are detected using 2-hybrid assays [98] and complexes [41]. Technology itself is advancing faster than the computational tools needed to analyze such data.

Different methods and technologies provide different viewpoints of the cells' activity: Gene expression data measures changes in the levels of mRNA, but fails to measure changes in levels of the active (e.g. phosphorylated) protein. Motif analysis on sequence promoters shows putative binding sites for transcription factors, but fails to identify neither the regulated genes nor the conditions under which active regulation occurs. Reconstructing biology requires the integration of many different data sources and technologies. Such integration not only provides a synergetic, multi-dimensional picture of the cells functioning, it also provides a means to detect and filter-out the noise in the given measurements. The biggest challenge is to develop models that fuse these different datatypes in a principled way. A number of attempts to integrate some of the above technologies have already been published [44, 80, 52, 85, 84, 102]. In addition, good automated experiment planning methods can greatly aid the goal of pathway reconstruction [53]. In this scenario the computational scientist will guide the experimentalist to perform experiments that are expected to give the largest amount of new information and gain [97].

The pathway reconstruction approaches discussed in this thesis were applied to *S. Cerevisiae*. While this is a big challenge in itself, one of the most important goals is to develop methods that would scale up to higher organisms. These are significantly more complex: consisting of more genes, many more layers of different types of regulation and communication between differentiated cells. While each model organism is quickly gaining enormous amounts of data, the real potential of data integration lies in the integration of data from different organisms. A known pathway in one organism can help reconstruct a homologous pathway in another organism. Such a comparative genomics approach has been proven successful in motif finding [60] and is the key to pathway reconstruction in more complex organisms such as mouse and human. In order to extend such an approach to pathway analysis, a formulation for a plausible model for pathway evolution is needed. Based on this model, a tractable distance measure between pathways can be devised. While holding great promise, there have been only a few initial attempts [24, 92].

Finally, the most exciting challenge of all, is to find ways to use the reconstructed models to better understand life itself. We can attempt to use these models to understand how biological regulation works, how cross talk between metabolic pathways and the environment can control levels of metabolites in the cell. Understanding the organization of genes into modules, and the higher order interaction between these modules can not only help design better reconstruction algorithms but also shed light on the robust multitasking workings of the cell. Once we understand the higher order organization of the cellular network, we could then ask how such an organization evolved. Finally, the biggest challenge would be to understand how failure of regulation leads to diseases such as cancer, and more importantly how to possibly fix such failures.

Bibliography

[1] S. Akutsu, T. Kuhara, O. Maruyama, and S. Minyano. Identification of gene regulatory networks by strategic gene disruptions and gene over-expressions. In *Proc. Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, 1998.

[2] B. Alberts, A. Johnson, J. Lewis, K.Roberts M. Raff, and P. Walter. *Molecular Biology of the Cell*. Garland, 2002.

[3] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Science*, 96(12):6745–50, 1999.

[4] O. Alter, P. Brown, and D. Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Science*, 100(6):3351–6, 2000.

[5] Z. Bar-Joseph, G.K. Gerber, D.K. Gifford, T.S. Jaakkola, and I. Simon. Continuous representations of time-series gene expression data. *Journal of Computational Biology*, 10(3-4):241–256, 2003.

[6] Y. Barash and N. Friedman. Context-specific Bayesian clustering for gene expression data. *Journal of Computational Biology*, 9(2):169–191, 2002.

[7] T. Blumenthal. Gene clusters and polycistronic transcription in eukaryotes. *BioEssays*, 20:480–487, 1998.

[8] H. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal of Computing*, 25(6):1305–1317, 1996.

[9] Breiman, Friedman, Olshen, and Stone. *Classification and Regression trees*. Chapman & Hall, 1993.

[10] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Monterey,CA, 1984.

[11] J. M. Cherry, C. Ball, K. Dolinski, S. Dwight, M. Harris, J. C. Matese, G. Sherlock, G. Binkley, H. Jin, S. Weng, and D. Botstein. *Saccharomyces* genome database. http://genome-www.stanford.edu/Saccharomyces/, 2001.

[12] D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*. Springer Verlag, 1996.

[13] D. M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 87–98. 1996.

[14] D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 80–89, 1997.

[15] G. Chisholm and TG. Cooper. Isolation and characterization of mutants that produce the allantoin-degrading enzymes constitutively in saccharomyces cerevisiae. *Mol Cell Biol.*, 2(9):1088–95, 1982.

[16] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[17] G. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 116–125. 1999.

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

[19] G.F. Cooper and C. Glymour. *Computation, Causation, and Discovery*. MIT Press, 1999.

[20] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2001.

[21] M.C. Costanzo, M.E. Crawford, J.E. Hirschman, J.E. Kranz, P. Olsen, L.S. Robertson, M.S. Skrzypek, B.R. Braun, K.L. Hopkins, P. Kondu, C. Lengieza, J.E. Lew-Smith, M. Tillberg, and J.I. Garrels. Ypd, pombepd, and wormpd: model organism volumes of the bioknowledge library, an integrated resource for protein information. *Nucleic Acids Research*, 2001.

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

[23] TS. Cunningham, R. Rai, and TG. Cooper. The level of dal80 expression down-regulates gata factor-mediated transcription in saccharomyces cerevisiae. *J Bacteriol.*, 182(23):6584–91, 2000.

[24] T. Dandekar, S. Schuster, B. Snel, M. Huynen, and P. Bork. Pathway alignment: application to the comparative analysis of glycolytic enzymes. *Biochem Journal*, 343:115–24, 1999.

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

[26] J.L. DeRisi, V.R. Iyer, and P.O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686, 1997.

[27] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mrna expression levels during cns development and injury. In *Pacific Symposium on Biocomputing*, pages 41–52, 1999.

[28] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[29] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, London, 1993.

[30] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Science*, 95:14863–14868, 1998.

[31] G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 144–151. 2001.

[32] G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: A structure-based approach. In *Proceedings of Neural Information Processing Systems (NIPS) 2000*, pages 479–485. 2000.

[33] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 421–460. Kluwer, 1998.

[34] N. Friedman, M. Goldszmidt, and A. Wyner. Data analysis with Bayesian networks: A bootstrap approach. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 206–215. 1999.

[35] N. Friedman and D. Koller. Being Bayesian about Bayesian network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–126, 2003.

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

[37] N. Friedman and I. Nachman. Gaussian process networks. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, 2000.

[38] N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 196–205. 1999.

[39] N. Friedman and Z. Yakhini. On the sample complexity of learning Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 274–282. 1996.

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

[41] AC. Gavin, M. Bosche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, JM. Rick, AM. Michon, CM. Cruciat, M. Remor, C. Hofert, M. Schelder, M. Brajenovic, H. Ruffner, A. Merino, K. Klein, M. Hudak, D. Dickson, T. Rudi, V. Gnau, A. Bauch, S. Bastuck, B. Huhse, C. Leutwein, MA. Heurtier, RR. Copley, A. Edelmann, E. Querfurth, V. Rybin, G. Drewes, M. Raida, T. Bouwmeester, P. Bork, B. Seraphin, B. Kuster, G. Neubauer, and G. Superti-Furga. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–7, 2002.

[42] D. Geiger and D. Heckerman. Learning gaussian networks. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 235–243. 1994.

[43] D. Geiger, T. Verma, and J. Pearl. d-separation: From theorems to algorithms. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 139–148. 1989.

[44] A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, and R.A. Young. Combining location and expression data for principled discovery of genetic regulatory networks. In *Pacific Symposium on Biocomputing*, pages 437–449, 2002.

[45] D. Heckerman and D. Geiger. Learning Bayesian networks: a unification for discrete and Gaussian domains. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 274–284. 1995.

[46] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 293–301. 1994.

[47] D. Heckerman, C. Meek, and G. Cooper. A Bayesian approach to causal discovery. In *Computation, Causation, and Discovery* [19], pages 141–166.

[48] T. Heinemeyer, X. Chen, H. Karas, A.E. Kel, O.V. Kel, I. Liebich, T. Meinhardt, I. Reuter, F. Schacherer, and E. Wingender. Expanding the TRANSFAC database towards an expert system of regulatory molecular mechanisms. *NAR*, 27:318–322, 1999.

[49] F. C. Holstege, E. G. Jennings, J. J. Wyrick, T. I. Lee, C. J. Hengartner, M. R. Green, T. R. Golub, E. S. Lander, and R. A. Young. Dissecting the regulatory circuitry of a eukaryotic genome. *Cell*, 95(5):717–28, 1998.

[50] NS. Holter, M. Mitra, A. Maritan, M. Cieplak, JR. Banavar, and NV. Fedoroff. Fundamental patterns underlying gene expression profile: Simplicity from complexity. *Proceedings of the National Academy of Science*, 97:8409–14, 2000.

[51] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–26, 2000.

[52] T. Ideker, O. Ozier, B. Schwikowski, and A. F. Siegel. Discovering regulatory and signaling circuits in molecular interaction networks. In *Proceedinds of the Tenth International Conference on Intelligent Systems for Molecular Biology.*, pages 233–240. 2002.

[53] T. Ideker, J.A. Thorsson, V. Ranish, R. Christmas, J. Buhler, J.K. Eng, R. Bumgarner, D.R. Goodlett, R. Aebarsold, and L. Hood. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 291:929–934, 2001.

[54] S. Imoto, T. Goto, and S. Miyano. Estimation of genetic networks and functional structures between genes by using bayesian networks and nonparametric regression. In *Pacific Symposium on Biocomputing*, pages 175–86, 2002.

[55] V. R. Iyer, M. B. Eisen, D. T. Ross, G. Schuler, T. Moore, J. C. F. Lee, J. M. Trent, L. M. Staudt, Jr. Hudson, J., M. S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P. O. Brown. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283(5398):83–7, 1999.

[56] J.D. Jaffe, H.C. Berg, and G.M. Church. Proteogenomic mapping reveals genomic structure and novel proteins undetected by computational algorithms. *Proteomics*, 2003.

[57] F. V. Jensen. *An introduction to Bayesian Networks*. University College London Press, London, 1996.

[58] S. Kalir, J. McClure, K. Pabbaraju, C. Southward, M. Ronen, S. Leibler, M.G. Surette, and U. Alon. Ordering genes in a flagella pathway by analysis of expression kinetics from living bacteria. *Science*, 292:2080–3, 2001.

[59] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The kegg databases at genomenet. *Nucleic Acids Research*, 30:42–46, 2002.

[60] M. Kellis, N. Patterson, M. Endrizzi, B. Birren, and E.S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423:241–54, 2003.

[61] FG. Kuruvilla, AF. Shamji, and SL. Schreiber. Carbon- and nitrogen-quality signaling to translation are mediated by distinct gata-type transcription factors. *Proceedings of the National Academy of Science*, 98(13):7283–8, 2001.

[62] 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, and R.A. Young. Transcriptional regulatory networks in *saccharomyces cerevisiae*. *Science*, 298:799–804, 2002.

[63] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. In *ACM Conference on Electronic Commerce*, pages 18–28, 2001.

[64] D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat Biotechnol*, 14(13):1675–80, 1996.

[65] G. MacBeath and S.L. Schreiber. Printing proteins as microarrays for high throughput function determination. *Science*, 289:1760–3, 2000.

[66] B. Mai and L. Breeden. Cln1 and its repression by xbp1 are important for efficient sporulation in budding yeast. *Mol Cell Biol.*, 20(2):478–87, 2000.

[67] S. Mangan, A. Zaslaver, and U. Alon. The coherent feedforward loop serves as a sign-sensitive delay element in transcription networks. *Journal of Molecular Biology*, 334(2):197–204, 2003.

[68] H.H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Science*, 94(3):814–9, 1997.

[69] HW. Mewes, K. Heumann, A. Kaps, K. Mayer, F. Pfeiffer, S. Stocker, and D. Frishman. MIPS: a database for protein sequences and complete genomes. *NAR*, 27:44:48, 1999.

[70] I. Nachman and N. Friedman. Inferring regulation kinetics from expression data. Technical report, The Hebrew University of Jerusalem, 2003.

[71] J. Norbeck and A. Blomberg. The level of camp-dependent protein kinase a activity strongly affects osmotolerance and osmo-instigated gene expression changes in *saccharomyces cerevisiae*. *Yeast*, 16:121–137, 2000.

[72] I.M. Ong, J.D. Glasner, and D. Page. Modelling regulatory pathways in *e. coli* from time series expression profiles. *Bioinformatics*, 18 Suppl 1:S241–S248, 2002.

[73] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[74] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge Univ. Press, 2000.

[75] J. Pearl and T. S. Verma. A theory of inferred causation. In *KR'91: Principles of Knowledge Representation and Reasoning*, pages 441–452. 1991.

[76] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17 Suppl 1:S215–S224, 2001.

[77] D. Pe'er, A. Regev, and A. Tanay. Minreg: Inferring an active regulator set. *Bioinformatics*, 18 Suppl 1:S258–S267, 2002.

[78] B. Raught, AC. Gingras, and N. Sonenberg. The target of rapamycin (tor) proteins. *Proceedings of the National Academy of Science*, 98(13):7037–44, 2001.

[79] C.J. Roberts, B. Nelson, M.J. Marton, R. Stoughton, M.R. Meyer, H.A. Bennett, Y.D. He, H. Dai, W.L. Walker, T.R. Hughes, M. Tyers, C. Boone, and S.H. Friend. Signaling and circuitry of multiple mapk pathways revealed by a matrix of global gene expression profiles. *Science*, 287:873–80, 2000.

[80] E. Segal, Y. Barash, I. Simon, N. Friedman, and D. Koller. From promoter sequence to expression: A probabilistic framekwork. In *RECOMB*, pages 263–272. 2002.

[81] E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module networks. In *Proceedings of the Ninteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*. 2003.

[82] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition specific regulators from gene expression data. *Nature Genetics*, 34:166 – 176, 2003.

[83] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(Suppl 1):S243–52, 2001.

[84] E. Segal, H. Wang, and D. Koller. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, 19 Supplement:S273–S282, 2003.

[85] E. Segal, R. Yelensky, and D. Koller. Genome-wide discovery of transcriptional modules from dna sequence and gene expression. *Bioinformatics*, 19 Supplement:S264–S272, 2003.

[86] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analisys. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 307–316, 2000.

[87] S.S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31(1):64–8, 2002.

[88] I. Simon, J. Barnett, N. Hannett, CT. Harbison, NJ. Rinaldi, TL. Volkert, JJ. Wyrick, J. Zeitlinger, DK. Gifford, TS. Jaakkola, and RA. Young. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 106:697–708, 2001.

[89] R. Somogyi, S. Fuhrman, M. Askenazi, and A. Wuensche. The gene expression matrix: Towards the extraction of genetic network architectures. In *The Second World Congress of Nonlinear Analysts (WCNA)*, 1996.

[90] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9(12):3273–97, 1998.

[91] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Number 81 in Lecture Notes in Statistics. Springer-Verlag, New York, 1993.

[92] J. Stuart, E. Segal, D. Koller, and S. Kim. A gene co-expression network for global discovery of conserved genetics modules. *Science*, 302(5643):249–55, 2003.

[93] A. Tanay and R. Shamir. Computational expansion of genetic networks. *Bioinformatics*, 17 Suppl 1:S270–S278, 2001.

[94] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nat Genet*, 22(3):281–5, 1999.

[95] H. Terashima, S. Fukuchi, K. Nakai, M. Arisawa, K. Hamada, N. Yabuki, and K. Kitada. Sequence-based approach for identification of cell wall proteins in saccharomyces cerevisiae. *Current Genetics*, 40:311–316, 2002.

[96] H. Terashima, N. Yabuki, M. Arisawa, K. Hamada, and K. Kitada. Up-regulation of genes encoding glycosylphosphatidylinositol (gpi)-attached proteins in response to cell wall damage caused by disruption of fks1 in saccharomyces cerevisiae. *Mol Gen Genet.*, 264:64–74, 2000.

[97] S. Tong and D. Koller. Active learning for structure in bayesian networks. In *International Joint Conference on Artificial Intelligence*, pages 863–869, 2001.

[98] P. Uetz, L. Giot, G. Cagney, TA. Mansfield, RS. Judson, JR. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and JM. Rothberg. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature*, 403:623–7, 2000.

[99] D. Weaver, C. Workman, and G. Stormo. Modeling regulatory networks with weight matrices. In *Pacific Symposium on Biocomputing*, pages 112–123, 1999.

[100] LF. Wu, TR. Hughes, AP. Davierwala, MD. Robinson, R. Stoughton, and SJ. Altschuler. Large-scale prediction of *saccharomyces cerevisiae* gene function using overlapping transcriptional clusters. *Nature Genetics*, 31:255–265, 2002.

[101] I. Xenarios, D.W. Rice, L. Salwinski, M.K. Baron, E.M. Marcotte, and D. Eisenberg. Dip: The database of interacting proteins. *Nucleic Acids Research*, 28(1):289–91, 2000.

[102] C. Yeang and T. Jaakkola. Physical network models and multi-source data integration. In *RECOMB*, pages 312–321. 2003.