Identification of Relevant Protein-Gene Associations by Integrating Gene Expression Data and Transcriptional Regulatory Networks

by

Angel Alvarez

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Chemical Engineering)
in The University of Michigan
2010

Doctoral Committee:

Assistant Professor Peter James Woolf, Co-Chair
Assistant Professor Xiaoxia Lin, Co-Chair
Professor Nicholas Kotov
Associate Professor Kerby A. Shedden

To my parents who always supported me,

To my son Angel Yamil who followed up my progress in a weekly basis,

To my friend Octavio Borges who told me 7 years ago that I could do this,

To my grandma Tita that even when she could not live to see this moment, she was always proud of her “soon to be a doctor” grandson

Table of Contents

List of Figures

Figure

List of Tables

Table

List of Appendices

Appendix

CHAPTER I

Introduction

One challenge in systems biology is integrating different biological data types to more accurately describe how a biological system functions. Most biological data types fall under one of the "-omics" classification such as *genomics* (DNA sequence, genome mapping, etc.), *proteomics* (protein structure and functions), *transcriptomics* (expression data, set of RNA's including mRNA's) and *metabolomics* (study of metabolites as a result of cellular processes). In this work I will show how gene expression data (transcriptomics) can be merged with gene transcriptional regulation networks (genomics and proteomics). By merging these data types, I demonstrate that I can identify which regulator-gene associations better explain the patterns of gene expression under different conditions and disease states.

I.1 Gene Expression Data

One of the most information rich and easily obtainable pieces of information about the state of the cell is a measure of its gene expression values. Gene expression is the process by which the information contained in genes is transcribed into an RNA

molecule (mRNA, tRNA, or rRNA) and eventually translated into a functional product, which in the case of mRNA is a protein. Knowing the level at which a specific gene is expressed in a particular cellular event provides information about which genes are involved in such event. For example, measuring the expression level of an oncogene in a tissue sample may indicate the presence of active cellular growth associated with a tumor.

A practical way to measure the expression of a gene is by detecting mRNA levels. Ideally the final product (a protein) is detected for the estimation of the gene expression too but this latest detection is more complicated due to other biological events that might affect the final active form of proteins. Northern blotting and reverse transcription quantitative polymerase chain reaction methods (RT-PCR and qPCR) are widely used methods to quantitatively measure mRNA levels [1, 2]. However, these methods can be time consuming, expensive and impractical when studying many genes in a sample.

A more cost effective way of measuring gene expression is to use microarray technology. Gene expression measurements using microarray technology allow to simultaneously measure the expression of thousands of genes. These simultaneous measurements are widely used for gene expression profiling in which a global picture of the cellular functions associated with the samples can be obtained. In a typical expression profiling analysis, a list of genes that shows difference in expression between two different types of samples can be identified statistically. In addition, methods such as Gene Set Enrichment Analysis (GSEA) [3] identify which differentially expressed genes in the samples are associated with known biological processes or pathways and suggest a more specific list of genes.

However, most current bioinformatic techniques only describe the pattern of differential expression without indicating the causal events that initiated the pattern. In this thesis, I attempt to fill this gap by integrating gene expression data with transcriptional regulatory networks.

I.2 Transcriptional Regulatory Networks

Another source of information showing how genes are regulated during the transcription process is regulatory networks. These networks contain transcriptional regulatory associations between proteins and genes. Transcriptional regulation is one of the earliest control mechanisms of gene expression. Interaction of regulatory proteins with DNA is the most direct method to change transcription levels. The transcription control responses are varied (enhancing, repressing, activating, etc.), but at the most basic level all of the mechanisms involve the binding of a protein to a regulatory binding site to exert its regulatory role on a gene.

Significant research has been invested to understand and predict these regulatory networks [4-6]. One common and used method to predict regulator-gene associations include sequence analysis on genes to identify specific regulatory binding sites or "motifs" where a known protein can bind. Databases such as TRANSFAC [7] contain information of proteins with their possible binding sites for eukaryotic genes regulation. Applications have been developed for this search and prediction process like for example Alibaba2 [8] among others.

Another method used to identify regulatory sites on genes is called phylogenetic footprinting in which different genomes are compared to identify conserved regions [9]. These conserved base sequences between genomes are again mapped to candidate regulators. All these regulatory associations predictions provide information about the global regulatory activities involved in different signaling. The main limitation using these transcription networks generation methods based on sequence analysis is that they also predict false positive associations. Binding sites are usually short (~5-15 bases), hence some of these predicted binding sites can be found by chance with no real regulatory function. Furthermore, some of these predictions, even when they are true, might not be active in some situations.

Another challenge associated with these regulatory associations, specifically with the active state of a regulatory protein, is that cells use other mechanisms to modify proteins after synthesis (post-translational modifications) and before they exert their regulatory role. As an example, Figure 1 shows how a group of 5 proteins (SET, HMG2, NME1, pp32 and Ape1) in a protein complex named "SET complex" await for the cleavage action of GZMA before each one can exert regulatory roles by binding to their respective DNA binding site. Cleavage is just one of others chemical modifications (phosphorylation, acetylation, etc.) that proteins frequently need before binding and regulate genes. Because of this regulatory complexity, one would not expect the mRNA level of a regulatory protein to map to the activity level of the protein itself. Indeed, research by others has shown that mRNA and protein expression levels only poorly correlate [10, 11].

The bioinformatic community has collected many of these regulator-gene relationships in databases such as RegulonDB for E.coli [12], MsigDB for human [3], among others. Different from gene expression data, these regulatory-gene networks provide the mechanism that, in part, drives changes in gene expression. However, even when a fully complete and accurate regulatory network is known, two main challenges remain that limit their study and interpretation: (1) The false positive predictions as explained above and (2) the challenge to infer which specific associations have a real regulatory role. As mentioned earlier, some regulator-gene activities will be only relevant for a given cellular environment. For example, some gene regulatory mechanism may only be used in cases of stress or during a short developmental stage. This same regulatory mechanism may not be relevant to other processes making a general regulatory network less useful for these cases.

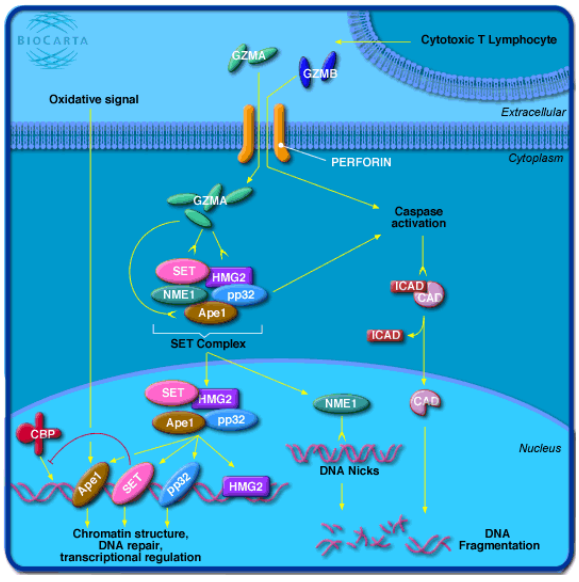


Figure 1. Example of protein-level regulatory processes influencing gene expression. http://www.biocarta.com/pathfiles/h_setPathway.asp. This diagram illustrates how a group of already synthesized proteins (SET Complex) are modified by protein level events before they actually bind and exert regulate roles. Cleavage by GZMA is the main modification to the group of proteins in the SET complex. Other proteins like Ape1 need additional cellular cues (oxidative signals). Measuring the activity of proteins is difficult but the collection of their binding sites is common. Databases containing these collections can be used to construct regulatory networks.

In this thesis, I hypothesize that merging gene expression data with transcriptional network information will allow me to identify relevant and possibly causal mechanisms

that govern the observed gene expression patterns. Due to the size of gene expression data sets and the complexity of regulatory networks information, a computational tool is needed to manage the merging task and further interesting queries about the system under study. The ideal computational tool should be able to handle graphical representation, noisy data, and any linear, nonlinear, and combinatoric relationships that might exist between regulatory proteins and genes.

I.3 Bayesian Networks

In the next sections I describe the machine-learning tool used in this project: Bayesian networks (BNs). First I will provide some information about Bayesian networks as they are the central tool used in this thesis. Next I will describe how BNs are uniquely well suited to the problem described in this thesis.

At its core, a Bayesian network is a directed graph that describes causal or apparently causal relationships between variables. For example, in this work, I adopted a bipartite graph representation to model a transcriptional network as shown in Figure 2. Variables in the bottom layer of the bipartite model represent gene expression profiles and the variables in the top layer represent protein activities. For computational efficiency, BN's analysis is mostly used on discrete variables specially if the amount of variables is large. In the regulatory model presented here, variables can be allowed to take 3 different outcomes such as low-med-high expression or activity level. Together, the variables have a joint probability that describes the probability of finding any set of

values for the variables. For example, for the network illustrated in Figure 2, the joint probability distribution can be written as:

$$P(R1, R2, G1, G2, G3, G4) = P(G1|R1)P(G2)P(G3|R1,R2)P(G4|R2)P(R1)P(R2)$$

This joint probability statement is a Bayesian network and it defines which variables influence which other variables and which variables are independent of each other. In terms of the graphical model, each variable is represented as a node and the arrows between nodes represent probabilistic dependencies indicating a causal relationship between the two variables. Note that the graphical representation of a Bayesian network is similar to that of a kinetic model such as a signaling pathway, but is interpreted differently. In a kinetic model, edges represent a specific function (activation, repression, a linear relationship, etc.), or a transformation (e.g. A -> B implies A becomes B). In a Bayesian network, these causal relationships may be any activation effect as well as inhibition and also includes linear, nonlinear, and/or multimodal associations between variables.

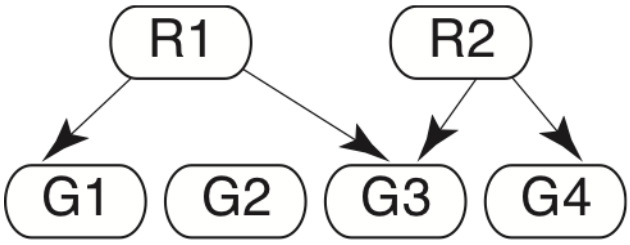


Figure 2. Bipartite model used to illustrate a transcriptional regulatory network.

Three features that make Bayesian networks analysis suitable for modeling transcriptional networks with gene expression data are (1) BNs are directed acyclic graphs (DAGs), (2) BNs can make quantitative predictions and (3) BNs are robust to noise and nonlinear inputs.

The first feature that makes BNs appropriate for this problem is the DAG property. With a DAG there is always a set of nodes that are ultimate causes and always a set of nodes that are ultimate responses. Because the main problem in this work is modeled as a bipartite network where only the top layer of regulators influence the bottom layer of genes with no allowed influences between regulators or between genes, the BN model studied here represent a simpler case of a typical Bayesian network. Because a Bayesian network is a statement of a conditional probability relationship, loops or cycles are not allowed in any Bayesian network. For example, consider a two variable system of A and B. If we were to create a model where A and B both influenced each other, we would obtain a model of the form P(A,B)=P(A|B) P(B|A) P(A|B)… The result would be a logical circularity and would not yield a working model or a clear statement of causation. A common question would then be "how do BNs handle biological loops?". A common way to handle loops is by using a time stamp on the variables allowing one variable in a time t_i to influence a variable in a time t_j even when it is the same variable or a loop is created if the time stamp is not present. For this, we require time series expression data that consist of periodic measurements of a sample to obtain a time varying gene expression profile.

In this thesis, I am using static expression data which consist of a set of expression measurements made under different conditions, treatments, or sample types. Nevertheless, feedback signaling between a protein and its gene can be present in the network because protein activity and the gene expression represent two different "entities" in our model. For example, protein activity of variable "A" can be regulating

the expression of variable “A” in the network without violating the DAG property of the model.

The second feature that makes BNs attractive for this study is that once a BN model is learned, it is possible to make quantitative predictions. The quantitative predictions produced by a Bayesian network have the advantage that they provide error estimates on the predictions in the form of probability distributions, and these predictions are directly comparable to experimental data. Using either approximate or exact inference algorithms, temporal predictions about how genes will respond to specific perturbations (genes knockdowns, mutations, etc.) can be made.

The third feature associated with BN’s robustness to noise and nonlinear inputs is a key element in making this computational tool attractive for this study. Because BNs are fundamentally probabilistic, they are well suited to handle noise and possible contradictions in data in a rational and systematic way. The main reason for this flexibility is that this approach does not assume an underlying analytical function to interrelate the variables in the model. BNs can model the linear, nonlinear, and combinatoric relationships present in biological data equally well.

There are different levels of difficulty when using and/or learning a BN depending on the type of data and the prior structural knowledge. In the next sections I explain the three general Bayesian network learning problems and the approaches used for each one.

I.3.1 Bayesian Network Learning: Complete data and known topology

When data are available and the best model describing the data is known, the Bayesian network learned is mostly used for inference purposes. The set of conditional probabilities for each variable (parameters of the model) is defined by the data and the model. The complete set of parameters for a BN model is referred as a conditional probability table (CPT). With this information, inference on the network can be done. For example, imagine that the model shown in Figure 2 is the best model explained by a discrete data set of binary variables that can take outcomes of "On" or "Off". Based on a hypothetical data set associated with the variables in the model, the conditional probability values for variable G1 is:

R1	P(G1=On | R1)	P(G1=Off | R1)
On	0.98	0.02
Off	0.30	0.70

Table 1. Hypothetical conditional probability values for variable G1

Using information from Table 1 the probability of finding variable G1 at any state is given by the state of its parent. For example, the probability of finding variable G1 "Off" given that its parent R1 is "On" is 0.02. More complex exact inference can be done using the information in CPTs and Bayes' rule, variable elimination methods or junction tree methods [13-15]. When the network is large and complex, reasonable probability estimates can be done using approximate methods such as Markov Chain Monte Carlo methods discussed in detail elsewhere [16].

I.3.2 Bayesian Network Learning: Complete data and unknown topology

When only a complete data set describing the variables in a system is available, Bayesian networks can be used to learn the most probable network explained by the data. In this case, the body of data is used to score candidate networks by estimating the probability of a network given the data set or p(network|data). To estimate the probability of a single network is difficult if the whole set of possible networks is not known. This problem can be simplified using Bayes' rule to rearrange the term p(network|data):

$$p(network \mid data) = \frac{p(data \mid network)\, p(network)}{p(data)} \tag{1}$$

The term *p(data)*, or the prior probability of the data, is a scaling term that does not change between candidate networks. The *p(network)*, or the prior probability of the network, is a term that can be used to favor certain networks based for example on expert knowledge or literature. When no information is available to favor any candidate network, a uniform distribution for all the candidate networks can be assumed. The term *p(data|network)* is the probability of the data given a network. This term is calculated by marginalizing over all parameter values (conditional probability values associated with each node). Note that this marginalization produces an automatic penalty for the more complex models, therefore favoring simpler graph structures [16]. A closed form solution for discrete data using standard approaches from probability theory exists for this term of *p(data|network)* [17, 18]. This solution is known as the Bayesian Dirichlet Equivalent metric (BDe) and has the following form:

$$p(data \mid network) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{2}$$

where "*n"* is the total number of variables, "q_i" is the total possible state configurations for a parent, "r_i" is the number of states of a variable (arity), "N_{ij}" is the number of cases parent of variable "*i"* is in state (or state combination) "*j*", "N_{ijk}" is the number of cases variable "*i*" is in state "*k*" and parent(s) in state "*j*". The expression in equation (2) describes the product of the probability of a variable being in a state *k* and the parents of this variable in a state *j*. The more informative the parents are of their child, the higher the value of *p(data|network)*.

With this metric available to score networks based on a body of data, it is possible to cycle through different networks configurations to identify which network is best described by the data. The challenge associated with this process is not calculating the score of a network but searching through the vast space of possible networks. Work by others has shown that exhaustive network search is an NP-complete problem [19], meaning that for all but the smallest problems an exact solution is not obtainable. To overcome this challenge, a wide range of tools from the field of discrete optimization such as greedy learning, simulated annealing, genetic algorithms, etc. can be used [16]. For example, in greedy learning, the learning process starts by choosing a network at random or an initial network created from prior knowledge known to have a mixture of correct and incorrect connections. The network is scored using the BDe metric. Next, a change is made to the network by adding, removing or reversing an arrow that does not violate the acyclic property of the network. After scoring the new network, if the score is better, the change is accepted and the network is subject to further modifications. If the score of the network is worst, the change is undone and a new modification is proposed. Note that in the case of a bipartite model like the one proposed in this work, only

connections removals or additions from the top layer towards the bottom layer would be allowed.

In the end, all network learning algorithms generate a ranked set of networks that are likely given the data. In some cases, the network with the highest score might be significantly better than all the other networks representing the best network described by the data. Alternatively, a group of the top-scoring networks can be very similar and a composite of those networks can be generated as a consensus network. Once a network or consensus network is learned from the data, it can be used to make inferences as described in section I.3.1.

I.3.3 Bayesian Network Learning: Incomplete data and unknown topology

Sometimes, besides not knowing the topology associated with the variables in a dataset, some pieces of data may be missing. Furthermore, there are cases where some variables in the network have no data at all. These variables are commonly referred as hidden variables. For example, as mentioned in section I.2, the activity values for regulators are difficult to measure and do not map well to the mRNA expression levels. In cases like the one presented in this thesis, the dataset is incomplete in the sense that only mRNA data are available for the variables in bottom layer of the network (genes mRNA expression) but no data are available for the top layer (regulators activity).

A modification on the network scoring process can be made to evaluate the score of a network even when some entries are missing. One approach to estimate an average score is by considering all possible missing entries state configurations. For example,

consider the network shown in Figure 3 where two regulators R1 and R2 are hidden variables. If these variables were binary (only having two possible outcomes), the total missing entries in a data set containing only 3 samples will be 6 (3 for each regulators). Therefore, the size of all possible missing entries state configurations will be $2^{\text{all missing entries}}=2^6=64$. An average score for this network can be estimated by averaging the scores obtained by using each of the 64 states configurations for the regulators. The limitation of this exact enumeration method is that is not computationally practical for large datasets having more than 2 or 3 hidden variables. For example, if for this same network in Figure 3 a 25 samples data set is available, the missing entries state configurations will be 2^{50} which is $> 1x10^{15}$!

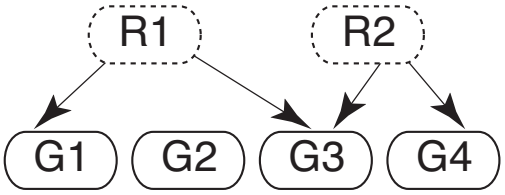


Figure 3. Model containing hidden variables. Here, the variables in solid ovals (genes) are observed and the variables in dashed ovals (regulators) are hidden.

An alternative to the exact enumeration approach to evaluate a score of a network in the presence of missing entries is adopting a sampling method to sample over the missing entries state configuration space. Gibbs sampling is a well-developed tool in computational statistics and has found extensive use in missing value estimation on Bayesian Networks [20-23]. Broadly, Gibbs sampling works in the following way:

- Values for all unobserved entries are randomly chosen each time a possible network is going to be scored using the BDe metric.

- A randomly chosen unobserved entry is re-sampled based on the probability distribution for that variable after estimating *p(network|data)* for each possible state.
- The new sampled value for an entry is used when evaluating a future entry.

The last two steps are repeated many times. Note that a Gibbs sampler does not select a single best data configuration, but instead samples a wide variety of possible configurations for the hidden values favoring the more likely configurations over the less likely ones. The result of this calculation is an average probability score of the network given the available data.

I.4 General research approach

In the following three chapters I will include a description of the modeling approach designed for the integration of gene expression data and general knowledge of transcriptional regulatory networks to predict relevant regulator-gene associations based on the data.

Because Bayesian networks are the computational tool used for this research work, in Chapter II I present a proof of concept analysis showing that Bayesian networks can recognize the most predictive regulator-gene associations even when no data are available for the regulators. I test a simplified toy model using both synthetic data and *E. coli* expression data to identify plausible regulatory modules.

In Chapter III, I introduce a novel approach I call POBN2 to modify the scoring method that allows the study of large networks with hidden variables. Synthetic data and *E. coli* gene expression data were used again to validate POBN2. The *E. coli* network studied in this chapter consisted of 189 genes and 62 hidden regulators.

In Chapter IV I shift focus toward identifying regulatory associations that are more likely to explain the gene variances. To carry out this analysis, I describe a new analysis tool I created called RegNetB. I test RegNetB using a human prostate cancer data set describing regulatory activities between 253 genes and 292 regulators.

Chapter V concludes this work by summarizing the capabilities of the designed approach and how it can aid in identifying potentially causal mechanisms for observed changes in gene expression.

CHAPTER II

Learning Regulatory Networks Using Gene Expression Data Alone

II.1 Background

The pathways that regulate mRNA expression play key roles in many cellular and disease processes. If these pathways were well characterized, then we could rationally design therapies to modify undesired gene expression patterns and the phenotypes that result from them. Unfortunately, most regulatory pathways remain only partially known. Here we present a new method for automatically reconstructing these regulatory pathways from data directly.

Bioinformatic models of regulatory pathways are generally constructed in one of three ways: (1) literature, (2) promoter analysis, and (3) analysis of mRNA expression data. These three methods are described graphically in Figure 4. Literature based regulatory pathways include those found in public databases such as KEGG, RegulonDB, and BioCarta. These pathways are hand-curated based on findings gleaned from a wide range of experiments and laboratories. While these literature based regulatory pathways are fairly accurate, they can only reveal transcriptional associations that have been explicitly studied, biasing the results toward well-characterized genes. Furthermore, these

literature-based pathways often use mRNA, protein, and protein activity interchangeably, producing models where the transcriptional regulatory steps are not clear.

A second approach to construct regulatory pathways is via promoter analysis. In this approach, the DNA sequence of the promoter region controlling the expression of each gene is analyzed to identify regulatory motifs that may be responsible for controlling mRNA expression. However, these regulatory motifs are usually short (~5-15 bases) compared to the whole genome, hence many of these predicted binding sites can be found by chance with no real regulatory function [24]. Thus, a disadvantage of promoter analysis is that many of the regulatory associations are false positive connections [8, 24, 25]. Furthermore, some regulatory activities are context specific, meaning that even when there is a binding site for a specific transcription factor, that regulatory mechanism may be inactivated by other proteins that don't bind directly to the promoter. An example of this off-promoter regulation is illustrated in the mechanism in Figure 4A, where the co-regulator (R2) regulates G3 by binding to its transcription factor (R1). For this case, the R1 binding site found in G3's promoter may or may not be active, depending on the activity of R2. A final, and possibly most significant challenge with promoter analysis is that promoters are often difficult to identify, particularly in eukaryotic organisms[26, 27].

A third approach is to use mRNA expression data directly to reconstruct the regulatory network. This data driven approach is attractive in that the most direct effect of a regulatory network is to control the mRNA expression levels. One commonly used method for analyzing these expression data to identify regulatory mechanisms is clustering, as is reviewed elsewhere [28-30]. Clustering identifies groups of genes that

have similar patterns of gene expression and therefore thought to be part of a common regulatory pathway. However, clustering as it is commonly carried out has two limitations. First, genes are generally clustered using a linear pairwise distance metric such as a Pearson's correlation coefficient. Gene regulation, in contrast, is largely nonlinear and combinatoric, meaning that most genes are regulated by a set of regulators that increase or decrease expression in non-additive ways. If two genes have identical regulation, then pairwise clustering will identify them as having a common regulatory mechanism (e.g. G1 and G2 in Figure 4C), while genes that have some similarities in their regulation may not cluster as well (e.g. G3 in Figure 4C).

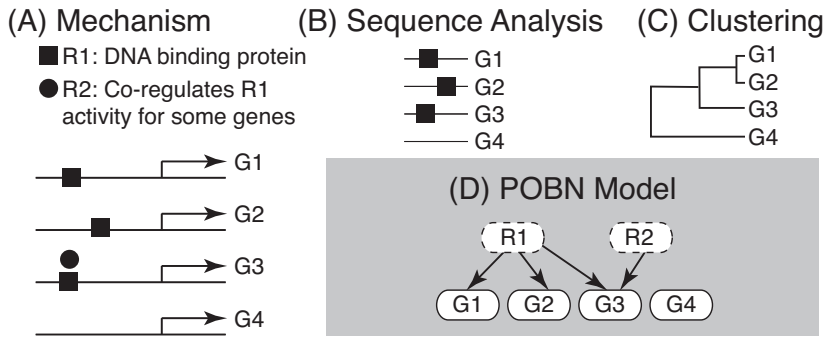


Figure 4. Methods for identifying regulatory mechanisms from gene expression data. (A) A hypothetical mechanism whereby two regulators (R1 and R2) control the expression of four genes (G1-G4). (B) View of this same system from the perspective of promoter sequence analysis assuming no false positive binding sites. (C) Expected clustering results for the expression data on G1-G4. (D) A partially observed bipartite network (POBN) model of the mechanism based on only the expression data for G1-G4.

A second problem with using clustering alone to find regulatory networks is that clustering does not suggest causation. A set of genes that tightly cluster may indeed have a common regulatory mechanism, but from the cluster one can't identify if one gene is responsible for the cluster or if an external factor governs the cluster.

In this work, we present a graphical modeling method to overcome some of the problems found in clustering to identify regulatory networks. The method we term Partially Observed Bipartite Network or POBN, uses a simplified Bayesian network topology to describe a regulatory network, as is illustrated in Figure 4D. A POBN has a top layer of unobserved regulators (protein activities) that connect to a lower level of observed variables (mRNA expression values). By casting the regulators as unobserved, a POBN makes it explicit that the activity of the regulatory proteins are not directly known. In some cases, the activity of a regulator could be simply proportional to the mRNA expression level of a transcription factor. Alternatively, the activity of a regulator could be the result of post-translational modifications, protein localization, or cleavage mediated by other processes. In this latter case, the activity of the regulator is difficult to directly measure, even though the effects of the regulator may be consistent. Related work using bipartite graph models have been used elsewhere to identify regulatory signals, but assume a linear mixing model [6, 31-34]. In contrast, the POBN models presented here use a multinomial model with Dirichlet priors, which has been shown to better reflect the nonlinear patterns seen in gene and protein regulation networks [35-38].

The POBN algorithm is tested using both synthetic data and experimental gene expression data from *E. coli*. *E. coli* gene expression data were used because this organism has one of the best studied gene regulatory networks available, allowing us to assess the quality of the predictions made by POBN.

II.2 Methods

In the following sections we describe the algorithms and sample studies used to evaluate the utility of POBN for analyzing gene expression data.

POBN Topology Scoring

Each network topology is modeled as a Bayesian network with the regulators as hidden nodes. In this network, variables are assumed to be discrete and modeled using a multinomial model with Dirichlet priors. The hidden regulatory nodes are scored using a Gibbs sampler to explore the large space of possible hidden node configurations [20-23]. The number of Gibbs sampling rounds required to accurately estimate the posterior probability density varies with the number of hidden nodes, quantity and quality of the data, and how the data are discretized. In this study, networks were scored using $1.4x10^7$ rounds of Gibbs sampling for the 266 sample case (532 missing entries in the dataset). This level of sampling was empirically found to be sufficient by repeating the scoring 10 times on the same dataset. At the end of these runs, the scores were nearly identical and the rank ordering of networks was identical.

Scoring was done using PEBL, a python library previously developed in our group [39]. PEBL evaluates the probability of a discretized dataset given a topology using the BDe scoring metric described elsewhere [17]. The source code for PEBL is freely available online (http://code.google.com/p/pebl-project/).

Network Searching

Because the networks used in this project are relatively small, we exhaustively

tested all unique topologies. In general, the full set of bipartite graphs with R regulators and G genes includes 2^{RG} networks. For 2 regulators and 4 genes used in this work this exhaustive set includes 256 possible networks. However, because the regulators are unobserved, many of these possible networks are structurally equivalent because the unobserved nodes can be permuted to yield identical predictions. For the two-regulator case, most networks have a structurally equivalent twin topology with R1 and R2 swapped. The exceptions to this twin rule are networks in which all hidden regulators share the same gene or group of genes. The total number of networks without a twin in the 2 regulators bipartite system is 2^G or16 for the 4 gene case. After accounting for these symmetries, the full set of unique 2 regulators 4 genes networks is 136. In the following case studies, all 136 unique network topologies are scored.

Consensus Network Evaluation

After all distinguishable networks are scored, they are sorted to identify a consensus regulatory network that captures features common to the top networks. By reporting a consensus network instead of the top network alone, we can provide a more robust estimate of topology predicted by the data. Unfortunately, construction of consensus networks is more complex when hidden nodes are present due to the structural equivalence of networks when the regulatory nodes are permuted. Thus the POBN consensus was identified using the following algorithm. First the networks from the top 99.9% of the normalized posterior density are included in the consensus network construction list. Next, the top scoring network topology is stored as the reference topology. Proceeding down the network construction list, each network is compared to the reference topology to determine which permutation of the regulator identities is most

similar to the reference topology. Similarity between topologies is measured by counting the number of edges in common. For each network in the construction list, we use the regulator permutation that is most similar to the reference. Finally, we constructed the consensus network by including only those edges that were present in every entry in the construction list.

In cases where a strong signal from the regulated variables is present in the data, the consensus and top scoring networks are the same. In contrast, if the signal from the regulated variables is weak, the consensus network will tend to be sparse.

Network Prediction using Synthetic Data

A synthetic test network was used to validate the network scoring and search algorithm when the underlying network topology is known. In addition, we used the synthetic network to evaluate the robustness of the scoring and search algorithm to noisy data, underlying nonlinearities, and different numbers of samples.

The topology of the synthetic network is shown in Figure 5. Values for the regulators in this model (R1 and R2) were randomly assigned from 0 to 1. The disconnected gene, G2, was assigned a random value between 0 and 1. The two genes with a single regulator were assigned using the linear models G1=1.5*R1, G4=0.5*R2. Gene 3 was assigned the maximum value of R1 and R2.

Given this model, sample datasets were created with 50, 266, and 532 observations. For comparison, the biological studies discussed later included 266 observations. After sampling, the values for the R1 and R2 were removed, making these

regulator variables unobserved. Next, all possible networks were scored to identify where the known and consensus networks appeared.

Data Discretization

The scoring metrics used in this study require that the data be discretized. In all cases in this paper, data were binned into two states, high and low, with the top half of the values assigned to high and the remainder to the low bin. This even sized binning strategy is widely used for discretization of gene expression data [36, 40], and has been shown empirically to be robust in capturing relevant details of the systems under study. The POBN method can be used with any discretization scheme and any number of bins, however a models with variables with many states will require significantly longer compute times to achieve convergence with the Gibbs sampler used by POBN.

Network Prediction from Gene Expression Data

We tested our algorithm using a set of 266 gene expression profiles in *E. coli* described elsewhere [41], and available online on GEO as GSE6836. This dataset represents a diverse range of biological backgrounds and environmental conditions including genetic perturbations, drug treatments, different growth phases, and a range of metabolic states. The study is well suited to this work because the large palate of perturbations should provide sufficient signal variation to detect meaningful bipartite regulatory networks.

To identify gene sets that could be meaningfully analyzed using our tools, we selected sets of 4 target genes that satisfied the following two criteria: (1) the target genes contained significant signal variation as measured by the variance of the gene's

expression across the chip; and (2) the 4 target genes had clear regulators based on the regulatory model presented in RegulonDB. The first criterion was further refined to limit the genes to only those in the top 200 most variation across chips to ensure that only the genes with the most signal were included in the analysis. These filtering steps ensured that the target genes had sufficient signal in this 266 observations study to be meaningfully evaluated. By selecting sets of 4 genes that have a clear model in RegulonDB, we could easily compare the results of our bipartite network search to a validated literature source.

After these steps, we found 3 networks that satisfied these criteria, shown in Figure 6 as Bacterial network I, II, and III. Note that larger networks could also have been selected, but due to computational limitations we only evaluated these models. For each set of 4 target genes, POBN was used to identify the score distribution for all two regulator/4-gene networks. In this step, the two regulators were assumed to be unobserved.

Next, the three bacterial networks were rescored assuming the regulators were observed. In this case, the observed regulators were inferred from the RegulonDB topology, and the activity of the regulator was assumed to be equivalent to the mRNA expression level of that gene.

II.3 Results

Network Prediction using Synthetic Data:

50 sample synthetic data set

The POBN results for three different samples of 50 data points showed the original network ranked in a position 27, 30 and 8 out of 136 models (see Figure 5A). The consensus ranked 2, 13, and 3. Interestingly, the scores of the networks divide into two tiers when only 50 observations are included, with this break most evident in Run 1 in Figure 5A. Examination of the networks in each tier indicate that the top tier contains combinations of features in the original network, while the bottom tier contains combinations of features that are not present in the original network.

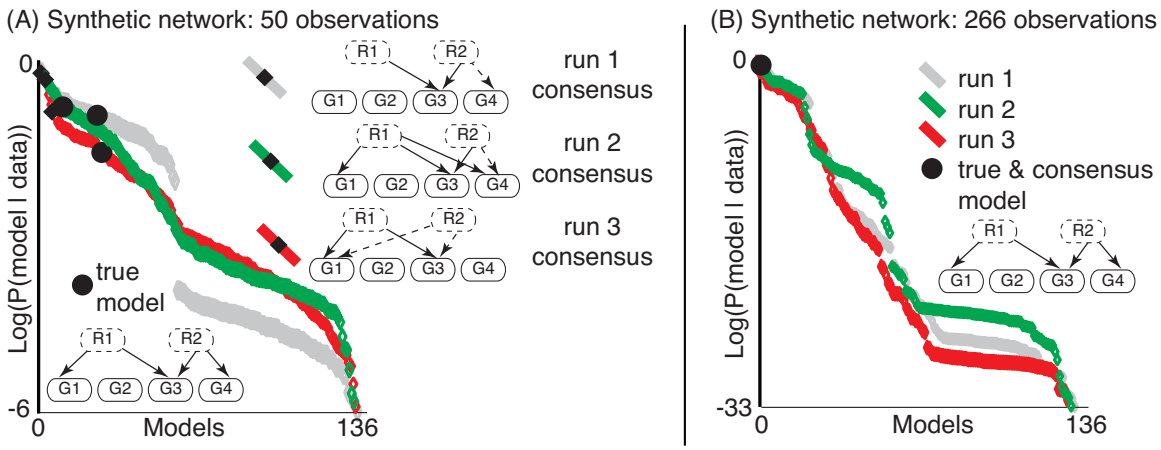


Figure 5: Synthetic network identification ability for different numbers of data observations. (A) Posterior distribution of models for 50 observations with the true model shown as a circle, and the consensus network shown as a diamond. Three different runs are shown, each one represents a different sampling of 50 observations. Dashed arrows indicate weakly supported edges, while solid arrows indicate strongly supported edges. (B) Posterior distribution of models for 266 observations. In this case, the true model and consensus model were at or near the top scoring models for each of the 3 different data samplings. These results indicate that POBN predictions with 266 observations are both accurate and stable.

266 and 532 sample synthetic data sets

When 266 samples were used, the true and consensus networks were the same and at or near the top of the distribution (Figure 5B). The tiers observed in the 50 sample case are also present when 266 samples are used, but there are more tiers when more data are included. For the 532 sample case, the true and consensus networks were also the top network by a larger margin. For computational efficiency, we only scored the top 25 networks found from the 266 sample case because of these topologies already captured most of the density in the previous rounds. These results indicate that that beyond a certain point, additional data for a given model does not yield useful increases in prediction accuracy. Furthermore, for this network size and complexity, a dataset of 266 samples is sufficient to accurately recover the underlying topology.

Network Prediction from Gene Expression Data:

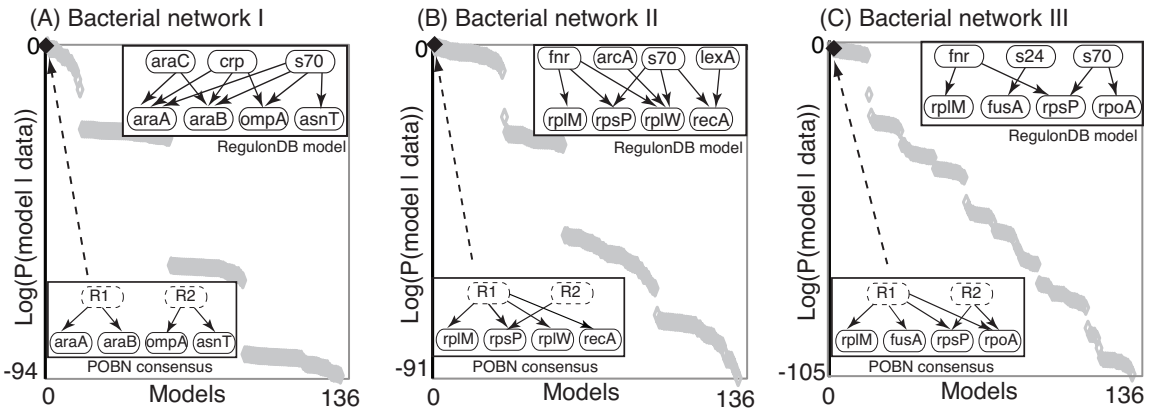


Figure 6: Regulatory modules found using *E. coli* gene expression data. Gray dots represent the sorted scores of each of the possible 136 POBN networks. The networks illustrated in the lower left corner of each panel represent the consensus network found for each case after POBN analysis. The networks at the upper right corner of each panel represent the RegulonDB models for the genes under study.

Data for three bacterial networks were scored in POBN, the results of which are shown in Figure 6. For each of these networks, the consensus network happened to also be the top-scoring network, and is shown as the POBN consensus model in the lower left of each panel. The complete network suggested by RegulonDB is also shown for each model in Figure 6. Interestingly, in all three cases, the posterior score distribution also contained clear separation between tiers, with Bacterial network I having only 4 tiers, while network III has at least 9 tiers. As can be seen by the complexity of the POBN consensus for each network, the more complex the consensus, the more tiers are present in the posterior.

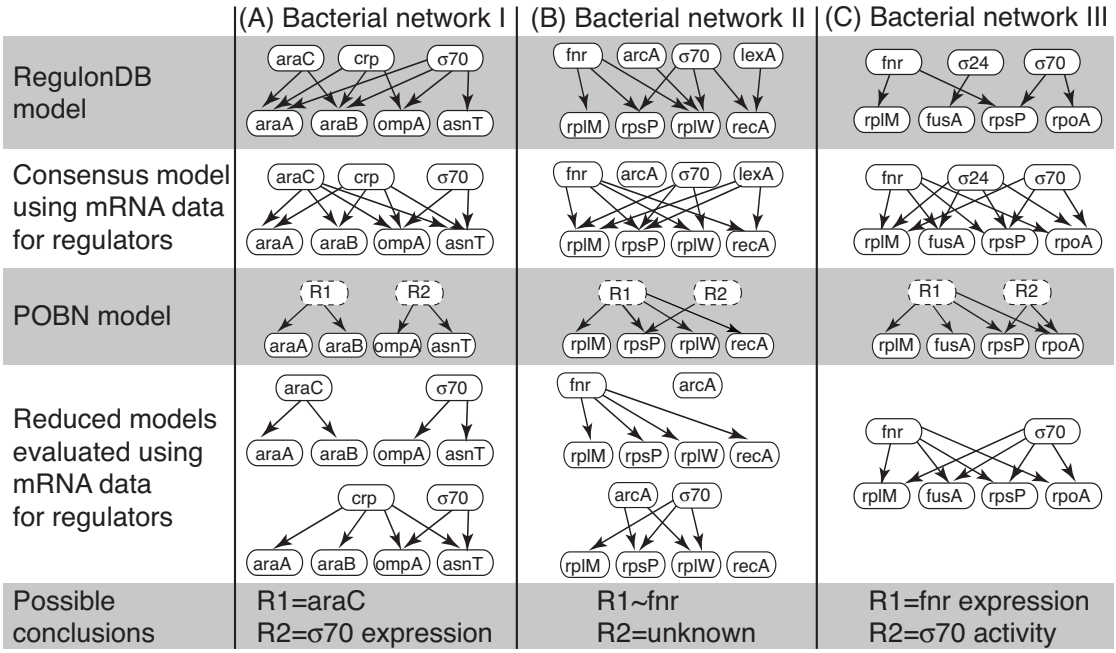


Figure 7. Network comparisons. Each column in the figure represents each of the biological cases analyzed in this study. The first row contains the models suggested by RegulonDB. The second row shows the consensus network found after scoring all possible models for the regulators/genes suggested by RegulonDB using mRNA data for the regulators. The third row shows the predicted models using POBN analysis (regulators treated as hidden variables). The second to last row show the consensus networks after exhaustively scoring all possible models for the variables shown and using mRNA data for the regulators. The last row summarizes the best estimate of the identity of the regulators based on the analysis.

To compare the RegulonDB models to the POBN models, we next rescored the networks assuming the regulators R1 and R2 were known. The results of this analysis are

shown in Figure 7. From these results, we see that in some cases the unobserved regulator can be fairly well predicted based on the expression values of the regulators suggested by RegulonDB. However, in Bacterial network II (Figure 7B), the POBN model is significantly different from any of the RegulonDB models, suggesting that other factors not captured by RegulonDB or by the mRNA data play a significant role in regulating these target genes.

II.4 Discussion

Synthetic data and general findings

Using a synthetic test case, we found that POBN was able to accurately identify the original model, or near the original model with surprisingly few observations as is shown in Figure 5. When only 50 observations were used, POBN recovered essential features of the original network, while with 266 or more observations, POBN was able to recover the original model exactly. Although this analysis does not provide a definitive estimate of the number of observations required for any POBN analysis, it does suggest that datasets on the order of hundreds of observations are sufficient to identify relevant features of a POBN type model.

The synthetic study also demonstrated that a POBN could capture nonlinear relationships. Because the POBN is based on a Bayesian network, we expected that both linear and nonlinear relationships should be detectable. In the synthetic network, target gene G3 was assigned the maximum value of R1 or R2, thereby creating a nonlinear relationship. The results in Figure 5 indicate that this nonlinear relationship was evident

with only 50 observations, and increasingly more evident with more data.

Tier formation

An unexpected finding was the appearance of tiers in the posterior plots shown in Figures 5 and 6. Intuitively, the top tier will group the models with the highest scores and the bottom tier the worst scored models. After an analysis of the connection weights between the regulators and genes, we always observed that the strongest and most significant connections (the ones shown in the consensus network) were present in each of the top tier networks. Similarly, the bottom tier networks consisted of networks with combinations of the weakest connections. In the middle tiers, there was a mixture of strong and weak edges. A clear example of this tier formation is the Bacterial network I illustrated in Figure 6(A). The four connections in the consensus network had an almost indistinguishable and high weight when compared to the other connections. Analysis of the networks in the second tier revealed that these networks all have R1 connected to G1 and G2. Similarly in the third tier, all networks have R2 connected to G3 and G4.

The presence of these tiers suggests a possible simplification to the POBN algorithm whereby only tiers of networks are identified instead of scoring each possible topology. In cases such as Figure 6A, a tier based search would only need to identify four roughly equivalent network families instead of the 136 searched in this study. Unfortunately, identifying the number of tiers *a priori* is still an open problem.

Identifying unknown regulators

Both an advantage and disadvantage to the POBN approach is that we don't know the identity of the variables in the unobserved regulatory layer. This lack of knowledge of

the regulators is an advantage in that it models a real case where a process that we may not know about governs the expression of a subset of genes. This unknown process could be the activity of another gene, a pathway, an environmental condition, or a particular physiological state of the cell. The clear disadvantage of this unknown regulator is that the regulator is unknown and therefore difficult to assess and test.

The unknown regulator problem arises in clustering in general, but we argue that the problem is less severe for a POBN model. When genes are clustered using a traditional method such as hierarchical clustering or k-means, it is often difficult to assign a common explanation or hidden regulator responsible for the cluster. In contrast however, POBN allows genes to be regulated by different numbers of regulators, thereby providing an implicit link between regulators. This link between regulators can be helpful for defining the regulator as it suggests a smaller set of possible causes.

Ways that the unknown regulator in a POBN model can be identified are illustrated in the analysis of the bacterial networks shown in Figure 7. In bacterial network I (Figure 6A and 7A), the POBN model has only marginal overlap with the RegulonDB model. The targets of R1 suggest that R1 could be araC, as they both have the same regulatory topology. When R1 was replaced with araC expression (Figure 7A, fourth row), the regulatory pattern remained, providing further evidence that R1 is araC. The targets of R2 predicted by the POBN model were most similar to the pattern of σ70, however RegulonDB also suggests that σ70 regulates araB and araA. When R2 is replaced by the expression of σ70, the regulatory pattern is identical to the POBN model.

In bacterial network II (Figures 6B and 7B) the POBN model and the RegulonDB

models are significantly different. A detailed analysis of the score distribution for this case suggests that recA is the least strongly connected to R1, providing weak evidence that R1 could be fnr. Furthermore, replacing R1 with the gene expression data for fnr produces model similar to the POBN model (Figure 7B, fourth row), which further suggests R1 could be fnr. The identity of R2 is less well defined. According to RegulonDB, σ70 coregulates rpsP, but σ70 also regulates rplW and recA—connections not observed in the POBN model. Furthermore, σ70 expression is not predictive of recA, but is predictive of rplM (Figure 7B, fourth row) in conflict with the accepted RegulonDB model. In this case, R2 appears to be a factor outside of the RegulonDB model.

In bacterial network III (Figure 6C and 7C) the POBN model and RegulonDB model disagree, but can be partially aligned to identify the regulators. Similar to bacterial case II, the POBN model for R1 is most similar to the fnr model in RegulonDB, except it is missing two connections. However, an examination of the literature reveals that fnr is known to regulate the rpsLG-fusA-tufA operon, which could explain the association between fnr and fusA [42]. Also, under certain environmental conditions the activity of fnr has been associated with rpoA responses [43, 44]. Furthermore, if R1 is replaced by the fnr expression level the regulatory pattern is the same as for the POBN model, suggesting that R1 is modeled well by the expression of fnr. Using the same logic, the POBN model for R2 is identical to the RegulonDB model for σ70. However, replacing R2 with the expression level of σ70 suggests that σ70 regulates rplM and fusA—a result that differs from the POBN and RegulonDB model. In this case, R2 may represent the activity of σ70, but not the expression level.

In all three bacterial networks it is possible that the two regulators are better described by something outside of RegulonDB. As an example, in bacterial networks II and III, it is possible that R1 is actually indicating the aerobic state of the cell, for which fnr is a readout. In cases like this, the identity of the regulator in the POBN model is ambiguous, but the pattern of regulation is clear.

Overall POBN provides an impartial and human interpretable method for identifying complex regulatory patterns directly from experimental data. In some cases, the identity of the regulators can be inferred using literature or other resources, however this identity need not be known to identify consistent regulatory patterns. Using a POBN model for identifying transcriptional regulatory networks provides a complementary approach to more traditional methods of transcription factor binding motif-based networks.

CHAPTER III

POBN2: Gene Expression Data and Transcriptional Networks Integration

III.1 Background

Significant effort has been invested in identifying which genes regulate the expression of which other genes in a given genome[4-6]. The bioinformatics community has collected many of these gene-gene regulatory relationships into transcriptional networks that provide a global view of how gene regulation is orchestrated. For example, TRANSFAC collects protein-DNA binding interactions to identify potential gene regulatory mechanisms[7]. Similarly, RegulonDB provides a hand annotated regulatory network for the *E. coli* genome[12]. As more data become available, these transcriptional regulatory networks will become increasingly complete in the sense that they will describe the set of possible mechanisms for regulating each gene.

However, even with a fully complete and accurate transcriptional regulatory network, only some of the regulatory relationships will be relevant for a given cellular environment. For example, some gene regulatory mechanisms may only be used in rare cases of stress, or during a short developmental stage. In these cases, these rarely used regulatory mechanisms are correct, but largely inactive and as such may not be relevant

to the process under study. In these specific cases, the general regulatory network is less useful.

In this chapter, we introduce a Bayesian network based method to differentiate active and inactive connections in a transcriptional regulatory network given a body of gene expression data. The method we term Partially Observed Bipartite Network 2, or POBN2, uses a simplified Bayesian network topology to describe a regulatory network, as is illustrated in Figure 2. A POBN2 has a top layer of unobserved regulators (protein activities) that connect to a lower level of observed variables (mRNA expression values). By casting the regulators as unobserved, a POBN2 makes it explicit that the activities of the regulatory proteins are unknown. As a first approximation, the activity of a regulator could be modeled as simply proportional to the mRNA expression level of a transcription factor, however this approximation ignores other regulatory events that are known to influence the regulatory process. For example, the activity of a regulator may be influenced by post-translational modifications, changes in protein localization, sequestration, and/or cleavage—all of which are mediated by other pathways in the cell. Unfortunately, this more complete view of transcriptional factor activity is complex, poorly understood, and difficult to quantitatively model. To circumvent this problem, the POBN2 approach allows the expression of the target genes to dictate the likely activities of each regulator. In doing so, POBN2 strives to identify regulatory topologies that are maximally consistent with both the expression data and the known regulatory network, while not specifying the mechanistic details that lead to the particular state of the regulatory proteins.

Expression data for learning these regulatory relationships can be divided into two

classes: time series and static. Time series data consist of periodic measurements of a sample to obtain a time varying gene expression profile, while static data consist of a set of expression measurements made under different conditions, treatments, or sample types. If designed correctly, a time series study can identify the sequence of events that trigger a regulatory event, as has been widely explored elsewhere[45-47]. In contrast, a static study can only be used to infer relationships between regulators without a clear picture of the sequence. Methods to use these static data for transcriptional regulatory network analysis have been less widely explored, although a majority of the gene expression data collected are static. For example, over 80% of the expression data in the public repository GEO are from static measurements. Although more challenging, in this work we have chosen to explore how these static data can be used to infer regulatory networks using POBN2.

To test the performance of the POBN2 algorithm for regulatory network reconstruction, we focus on the regulatory networks in *E. coli.* The regulatory network in *E. coli* is one of the best characterized, giving us a clear picture of the network we expect to find. In addition, digital resources such as the RegulonDB database provide the regulatory network in a machine readable form that is suitable for comparison to the POBN2 results.

III.2 Methods

In the following sections we describe the algorithms and sample studies used to evaluate the POBN2 algorithm's ability to identify active and inactive regulatory

relationships based on gene expression data.

Gene expression data

We tested POBN2 using a set of 266 gene expression profiles in *E. coli* described elsewhere [41], and available online on GEO as GSE6836. This dataset represents a diverse range of biological backgrounds and environmental conditions including genetic perturbations, drug treatments, different growth phases, and a range of metabolic states. The study is well suited to this work because it contains many samples and covers a range of perturbations.

The selection of genes that could be meaningfully analyzed using POBN2 was based on the following two criteria: (1) genes must exhibit differential expression in at least some samples; and (2) genes must be present in the RegulonDB regulatory network. The first criterion was enforced by selecting the 300 genes with the largest variation as measured by the magnitude of the standard deviation of the expression value across samples. This selection approach will tend to favor genes with larger absolute expression levels, and a diverse range of expression values across the samples. When the second criterion was applied, only 189 of the 300 genes were found in the RegulonDB network, producing a final list of 189 genes.

Data discretization

For computational efficiency, the scoring metrics used in this study require that the data be discretized. Data were binned into three states, high, medium and low, with the top third of the values assigned to high, the bottom third assigned to low, and the remaining values assigned to the medium bin. This even sized binning strategy is widely

used for discretization of gene expression data in the systems biology community [36, 40, 48, 49], and has been shown empirically to be robust in capturing relevant details of the systems under study. We note that the POBN2 method can be used with continuous values directly, however the continuous scoring algorithms that are currently available are computationally impractical for networks involving more than a few hidden nodes. Instead, POBN2 scores the probability of each network using discrete data as described below.

Scoring method

The regulatory bipartite networks tested here are modeled as a Bayesian network with the regulators as hidden nodes. In this network, variables are modeled using a multinomial model with Dirichlet priors, as is described elsewhere [17, 38, 50]. See section I.3.3 for a more detailed discussion about Bayesian networks with hidden nodes. Below we provide a brief summary of the method.

In most Bayesian network problems, a completely observed data set is used to estimate the likelihood of a network. However, in the network used in this work, the activity of the regulators is not observed. To fill in the missing activity levels of the regulators, the hidden regulatory node state values were estimated using a Gibbs sampler to sample the space of possible hidden node configurations [20-23]. The sampling and scoring was done using PEBL, a python library previously developed in our group [39]. PEBL evaluates the probability of a discretized dataset given a topology using the BDe (Bayesian Dirichlet equivalent) scoring metric described elsewhere [17]. The source code for PEBL is freely available online (http://code.google.com/p/pebl-project/).

The scoring process in our simulations comprised two main steps: (1) Gibbs sampling of the hidden variables (after burn in) using the initial global graph and (2) calculating of an average score across samples. The score for each network was estimated using the BDe metric and the sampled states for the hidden nodes. This scoring process was carried out for the initial network and each single edge removal from that initial network.

The number of Gibbs sampling rounds used to estimate the score for any network was determined empirically. The threshold used to cast a connection as active or inactive was the rank of the initial global network based on score relative to all the other tested networks as will be explained in the next section. Based on this approach, Gibbs sampling round sizes of 100, 250, 500 and 1,000 samples were tested in the first POBN2 optimization round. We observed insignificant changes between the results obtained for 250, 500, and 1,000 rounds of sampling. For these 3 sample sizes, 3 inactive edges out of 75-78 cast as inactive (as described in the next section) were observed to be different between each set results. Nevertheless, the networks evaluated after disconnecting these three discrepant edges ranked just below the threshold established to cast the edge as active/inactive. For the 100 Gibbs sampling rounds case, the difference was of 6 edges. As a result of this analysis, all subsequent POBN2 optimization rounds were run using 250 Gibbs sampling rounds as a balance between accuracy and computational efficiency.

Network searching and optimization process

The filtered 189 genes were mapped to target genes in the RegulonDB network. This mapped network produced a bipartite graph between regulators and targets

containing 570 connections between 62 unobserved regulators and the selected 189 genes. Given this network as an initial global network, we examined all one-edge removals in this network, resulting in 570 additional networks, for a total of 571 networks.

If removing an edge in the global network improved the score then the disconnected edge was labeled as inactive. If the initial global network did not rank first on the list, we proceeded with another scoring round after updating the initial global graph by eliminating the set of inactive connections. These steps were repeated until the starting global regulatory network (initial network for each round) ranked first on the list as the best network. Figure 8 shows a conceptual flow diagram of this process. Source code for the optimization is provided in the Appendix 1.

Synthetic Network Validation

To assess the POBN2 algorithm's discriminatory capability, we tested the algorithm using a synthetic dataset where the active and inactive edges are known beforehand. First, a bipartite network was defined with 4 regulators and 9 response variables. Each response variable was assigned to have 1-3 parents for a total of 11 connections in the synthetic network. After defining the topology, conditional probabilities were assigned for all nodes based on connectivity and the 3 discrete values that were allowed. Based on these parameters, 5,000 samples were generated to create a data set simulating a discrete static data set. Full details of the data generation script are provided in the Appendix 2.

After sampling, the values of the 4 regulators were removed, making these regulator variables unobserved. Next, all combinations of possible inactive connections were added to the initial graph one at a time for a total of 25 cases, i.e. 25= (4 regulators)*(9 response variables)-(11 defined connections). Once a single inactive edge was added, the weight for all edges in the graph were evaluated as described in the Scoring Method section above. This same process was repeated for all combinations of pairs of inactive edges that could be added to the original graph (300 pair addition cases).

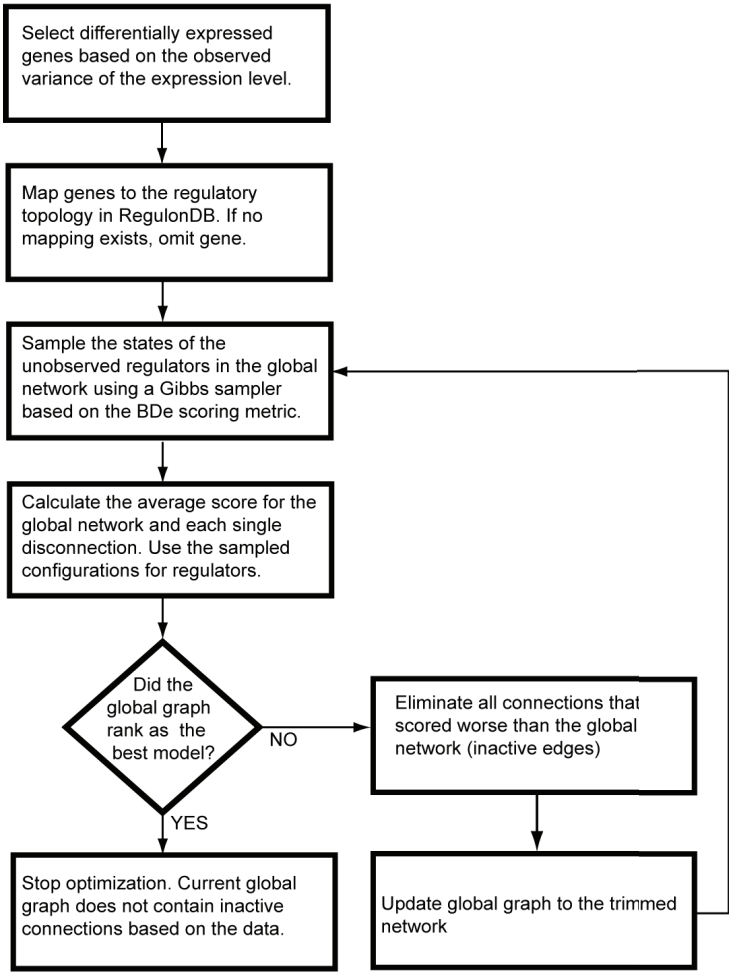


Figure 8: POBN2 algorithm conceptual diagram**.** Note that the creation of the inactive edges list is based on the rank of the network after an optimization round. If a connection is eliminated and the resulting model has a worse score than the initial network, this connection should stay as it plays an explanatory role based on the data. Otherwise, the connection is listed as inactive and eliminated from the global network.

III.3 Results

Synthetic data set

The synthetic case study provides a way to evaluate POBN2's ability to discriminate between inactive and active edges with a known network. When a single inactive edge was added in all possible positions, 23-22-24 out of 25 inactive edges were correctly identified in three independent runs, and in no case was a true edge called inactive. For the cases where POBN2 did not call the added edge inactive, the inactive edge was ranked as the first or second weakest (score just below the initial graph). When a pair of inactive edges were added in all possible positions, in 282 out of 300 cases the inactive edges were correctly identified and in 13 of the remaining 18 cases the connections were ranked as the first or second weakest. Here again, in no case was a true edge called inactive. In runs with fewer samples the number of hits were lower but the algorithm was consistently precise in not calling a true edge inactive.

Different levels of false edges present in the initial network were also tested keeping the samples size constant (5000 samples): A fully connected graph with 100% of all possible false edges present, 80%, 60%, 25% and 10% of all possible false edges were tested. Results for these tests showed that POBN2 started identifying the false connections in the graph when the level of false edges present was between 25-30% of the total possible false edges. Again, in none of the cases POBN cast a true edge as an inactive one.

Overall, these synthetic results indicate that the POBN2 algorithm is frequently able to correctly identify the added inactive edge(s) with no observable false positive rate

and a moderate false negative rate. Also, it indicates that the quality of the initial network plays an important role in the ability of POBN2 to identify the inactive edges. This result suggests that the edges POBN2 identifies as inactive in the biological dataset will likely be inactive, however, edges deemed active may be only marginally so, depending on where they appear relative to the initial network.

Inactive connections predicted using gene expression data

When the POBN2 analysis was applied to experimentally gathered gene expression data, the algorithm identified 93 inactive connections out of the 570 connections present in the initial regulatory network from RegulonDB. From this total, 75 inactive connections were found during the first score-ranking round of POBN2 analysis, 13 in a second round, and the last 5 in a third round. After the third optimization round, the score of the initial graph was no longer improved by removing any edges. During all the rounds of optimization, we observed that well characterized regulatory associations in *E. coli* such as the genes regulated by LexA (recA, recN, sulA, umuD, lexA) and AraC (araA, araB) were within the group of the best scoring edges in the network. A complete list of the 93 connections cast as inactive is provided in Table 2.

Regulator(s)	Gene(s)
CpxR, EnvY, Lrp, OmpR	ompC, ompF
FruR, IclR	aceB
GutM, GutR	srlA
Rob, SoxS	sodA
GalR, GalS	mglB, galE, galT
Fis	nuoB, nuoE, hupA
CRP	cyoA, cyoB, ompF
FNR	sucB, sucC, sdhA, sdhD, sdhC
ArgP	nrdB, nrdA
ArgR	nusA, metY
HU	galE, galT
ArcA	mdh, nuoB, sucB, sucC, aceB, sdhA, sdhD, treB, sdhC
CspA	Hns
NarL	nuoB, nuoE
RstA	ompF
TorR	hdeB
Sigma70	serV, metW, metZ, metV
Fur	sodB, cyoA, cyoB, ompF, sodA
FlhDC	mglB, mdh
DgsA	ptsH
Sigma38	mglB, galE, galT, hdeB, hdeA
CdaR	rnpB
Sigma32	gapA
MarA	sodA, hdeB, hdeA
H-NS	sodB, srlA, galE, galT, cydB
GadE	cyoA, cyoB, hdeB
GadX	hns, hdeB
TreR	treB
IHF	sucB, sucC, nuoB, sodB, ompC, nuoE, aceB, ompF, soda

Table 2. List of connections removed after POBN2 optimization. Regulators in first column were initially connected with the corresponding genes in the second column.

III.4 Discussion

By integrating a specific expression data set and a global regulatory network, POBN is able to identify a simplified regulatory network that is both mechanistically sound and maximally consistent with the expression data. This simplified network suggests which connections are of particular importance in the expression data.

During the optimization process, we observed that POBN2 tended to remove edges from genes that had the higher connectivity (>3 parents). In the initial regulatory

network from RegulonDB, each target gene had between 1-9 parent regulators. After POBN2 optimization, the connectivity range was between 1-6 regulators per gene with most genes having between 1-4 parents.

An extreme example of regulator trimming took place for the gene ompF. This gene is associated with 9 regulators based on the RegulonDB database. After optimization, 8 regulatory connections were removed from ompF suggesting that, based on the data set under study, the expression of this gene is better explained by 1 regulator rather than 9. Upon examination of the siblings of ompF in the optimized network (e.g. the genes that are also regulated by the same single parent), we see a regular, but nonlinear relationship between the gene expression values of ompF and its siblings (see Figure 9).

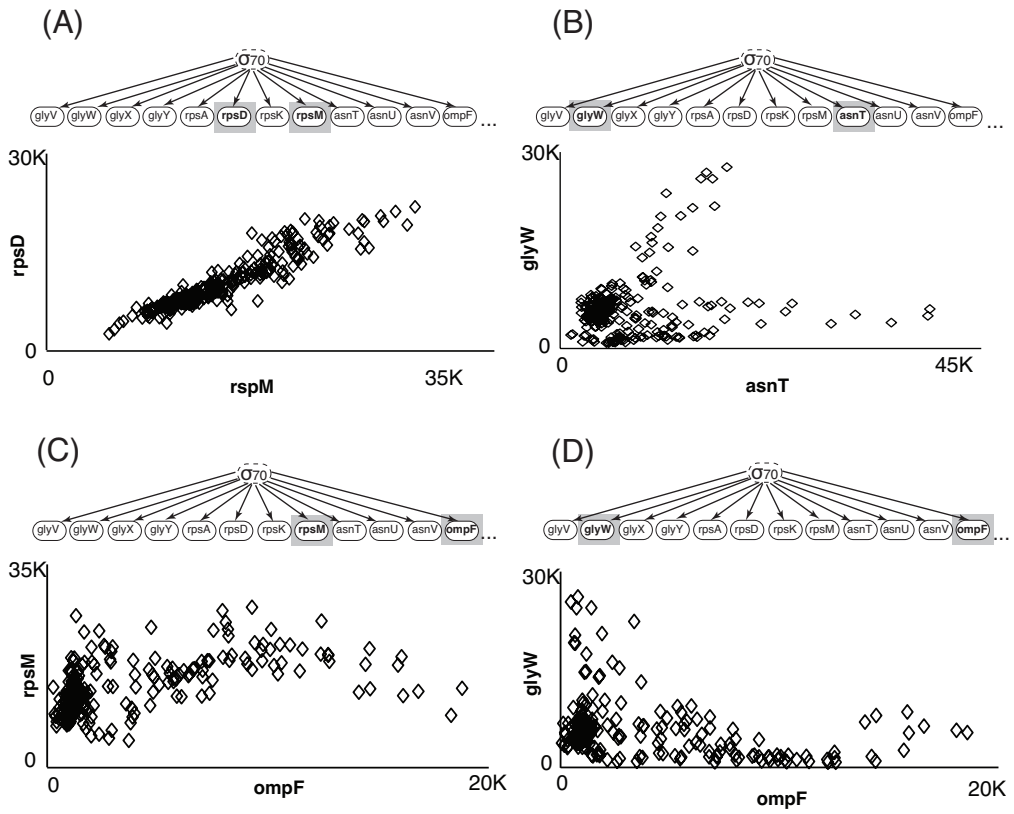


Figure 9. Expression value comparisons between siblings. All the genes shown are regulated by the same single regulator Sigma70. Panels (A) and (B) compare the expression values of genes that originally appeared regulated by the same regulator in the global regulatory network. The genes rspD and rspM shown in (A), are nearly linear related, while glyW and asnT shown in (B) have a more complex relationship. After POBN optimization, ompF gene was disconnected from 8 of its 9 originally regulators in the global regulatory network. Panels (C) and (D) illustrate two examples of the association of ompF with genes under this same single regulator. Note that different but clear non-linear associations are observed between ompF and its siblings.

One may ask why any edges should be removed from the annotated regulatory network at all. Presumably the annotated network is well validated experimentally, and as such should represent our best prediction of the gene regulatory relationships. Based on the analysis here, we see two possible explanations. First, it is possible that the edge identified as inactive is not actually a regulatory relationship at all. In this case, the

regulatory relationship present in RegulonDB would be an error or missannotation in the database. A second more likely reason for an edge to be called inactive is that the regulatory association is biologically correct, but is not active in the gene expression samples used in this study. In this latter case, removing the regulatory connection does not suggest that the there is no biological mechanism for the relationship, but instead that the connection is not relevant to the study. In both cases, POBN2 will favor reducing the number of regulators for each gene and consequently reducing the complexity of the model with little or no impact on the model's ability to predict the observed gene expression data.

The genes metZ, metW, and metV illustrate an interesting ability of POBN2, as all of these genes are associated with only Sigma70 in the prior global network from RegulonDB. After optimization with POBN2, these methionine t-RNA coding genes ended with no parents at the end of the analysis as shown in Figure 10 (C). A possible explanation for this parent elimination is that, for these genes, the prior network in RegulonDB did not have a complete list of possible regulators. There is evidence in the literature that under grow rate perturbations, the factor for inversion stimulation Fis, is been known to drastically alter the tRNAs pool composition including methionine tRNAs [51]. The intracellular concentration of this global regulator (Fis) varies substantially in response to changes in the nutritional environment and growth phases [52], conditions present as part of the samples set used in this study. It is possible that the expression variance for these genes metZ, metW and metV is better explained by a different regulator, i.e Fis instead of Sigma70. These results suggest that even when the true regulator is missing, POBN2 can still discriminate between consistent and inconsistent

connections.

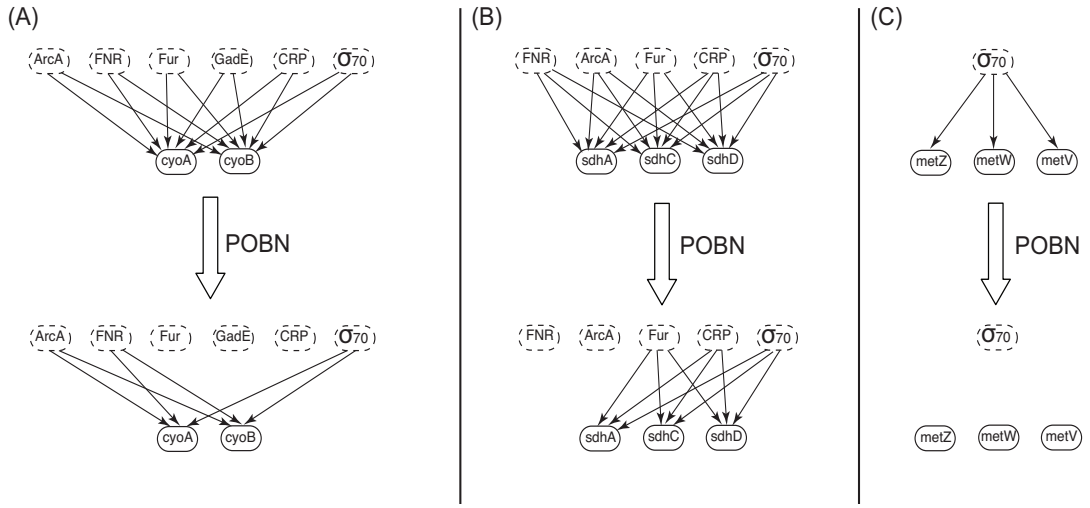


Figure 10. Optimization of genes regulated by the same group of regulators. The networks in the top section are sub-networks of the original global network. Each sub-network has the original connectivity for the genes as suggested by RegulonDB. In each case, each gene has an identical parent set. Note that in the three cases, the same inactive connections were predicted for each gene within each group.

By using a Bayesian network based approach the POBN algorithm is able to identify both linear and nonlinear relationships. For example, the regulatory relationship shown in Figure 3 includes both linear and nonlinear relationships. By identifying both kinds of relationships, POBN is able to detect relationships between genes that are not possible to detect using linear methods or most commonly used clustering methods.

In a larger context, the POBN approach provides a general way to integrate static observational data with knowledge about known regulatory relationships. In the example provided here, we found networks that were maximally consistent with both a set of gene expression data and a gene regulatory network. One could use a similar approach to identify relevant or active signaling pathways or protein phosphorylation networks from a mixture of experimental data and known topologies. By using POBN, all

members of the network need not be directly observed, as long the measurements that are used in some way reflect the activity of the unobserved nodes.

CHAPTER IV

RegNetB: Predicting Explanatory Regulator-Gene Relationships

IV.1 Background

What changes are responsible for making a tumor a tumor? If we knew the underlying cause for this change, then it may be possible to directly address the underlying dysfunction that causes tumorogenesis. One possible route to identifying a causal mechanism for tumorogensis is to gather a rich body of experimental data describing the state of many tumors and search for relevant signatures. Unfortunately, it is difficult to distinguish the signatures that are a consequence of the dysfunction from the signatures that cause the dysfunction.

A further complication is that the activity of the factors that influence gene expression is difficult to observe directly. For example, consider the simplest case of a single transcription factor that regulates the expression of one target gene. In this case, the activity of the transcription factor may be governed by its past history of mRNA expression, possible splice variants, protein modification, binding with other factors, and where the transcription factor is localized in the cell. In this case, the most direct measure of the activity of the transcription factor is the expression of the target gene itself. However, when multiple genes are coordinately regulated by multiple regulators,

analyzing these cause and effect relationships becomes more difficult.

One source of information relating transcription factors to their target genes is the transcription factor-DNA binding information in databases such as TRANSFAC and MsigDB [3, 7]. However, knowing transcription factor-DNA binding relationships alone does not identify which regulatory activities are relevant for a specific disease or tissue under study [24, 25]. This limitation can be partially overcome if gene expression data are integrated with transcription factor-DNA binding information to identify which transcriptional activities better explain the observed expression variation.

Regulatory Networks-Bayesian (RegNetB)

In this work, we have developed and tested a tool called Regulatory Networks-Bayesian, or RegNetB, to carry out this integration of gene expression data and transcription factor-DNA binding information. RegNetB uses a simplified topology to describe a regulatory network in which the top layer of this network represents the group of unobserved regulators (transcription factor activities) and the bottom layer represents observed genes (mRNA expression values). This regulatory bipartite network model has been used elsewhere to represent transcriptional regulatory networks by adopting a linear mixing model [6, 31, 53]. Here we extend these models to account for nonlinear and combinatoric effects using a multinomial Bayesian model with Dirichlet priors as described elsewhere [17, 38, 50].

RegNetB is tested using gene expression data from a prostate cancer study carried out elsewhere [54, 55]. Despite the high incidence and mortality rate, the molecular mechanisms underlying the oncogenesis and progression of prostate cancer are still

unclear. Significant research has been dedicated to identifying prognostic markers, however less research has focused on identifying the regulatory mechanism that drives the disease [56].

By identifying a group of the most relevant regulatory relationships, RegNetB is able to identify which regulators are most likely responsible for the expression variations in the prostate cancer study evaluated here. In the next sections we describe the data processing and results obtained after RegNetB analysis.

IV.2 Methods

In the following section, I describe the RegNetB algorithm and the data preprocessing used in our test cases.

RegNetB algorithm

The transcription factor-gene network presented here is modeled as a Bayesian network by RegNetB. Regulators in this network are modeled as hidden variables and the observed variables (genes) are modeled using a multinomial model with Dirichlet priors as described elsewhere [17, 38, 50]. Below we provide a summary of the scoring process.

For a typical Bayesian network scoring problem, a complete discrete data set describing the variables included in the network of interest is available. However, in this case again the transcription factors are not observed. To fill in the activity levels for the regulators, a Gibbs sampler is used to sample over the space unobserved regulators [20-23]. Gibb's sampling and network scoring were carried out using PEBL, a python library

developed in our research group [57]. PEBL estimates the probability of a discretized dataset given a specific network using a Bayesian Dirichlet equivalent metric described elsewhere [17]. The source code of PEBL can be freely downloaded from (http://code.google.com/p/pebl-project/).

Two scoring steps are performed by RegNetB to evaluate the relative strength of each connection in the transcription factor-gene network. First, sample states of the unobserved transcription factors are taken using a Gibbs sampler. The sample states are taken after a burn in of 10 iterations. The second scoring step uses these sample states to rescore the whole network when each transcription factor-gene edge is removed and then re-added. The relative importance of the edge can then be interpreted as the change in the average score of the network when the edge is removed versus present.

To generate the final list of regulators and genes of interest in our study, we first ranked all the connections based on the scores estimated by RegNetB. After normalizing all the connection scores, a graphical analysis was used to identify thresholds that differentiate a group of relatively stronger connections from the rest based on their scores. A list of all the genes and regulators was generated from this set of connections.

Global human transcription factor-gene network

A global human transcription factor-gene network was created using the Molecular Signatures Database (MsigDB) [3]. The source of the "C3: Motif Gene Set" information in this database, the collection we used to create the global human transcription network, is described elsewhere [9]. Briefly, the transcription factor binding sites were predicted using promoter sequence analysis, gene set enrichment analysis

(GSEA), and comparative genomic analysis. After collecting these transcription factor binding sites and the genes associated with them, the gene names were mapped to their official Entrez gene symbols. Only those genes mapping to unique official gene symbols were included. Similarly, some binding sites mapped to known transcription factors (regulators) names documented in TRANSFAC while others were only described as the sequence of the promoter itself. Regulatory sequences not mapping to any known regulator were listed as UK (unknown) followed by an integer.

Gene expression data

We used RegNetB to analyze 146 gene expression profiles from prostate tissue samples described elsewhere [54, 55] and available online on GEO as GDS2545. This set of expression profiles includes 18 profiles from normal prostate tissues, 63 profiles from normal prostate tissues adjacent to localized tumor, and 65 profiles from primary prostate cancer tumors. The 146 gene expression profiles were pre-processed using the web-based genechip analysis system (WGAS) described elsewhere [58, 59] for data normalization and mapping of probe sets ID to official gene symbols.

Next we filtered the gene list to only include genes that could be meaningfully analyzed. The genes passing the filter must: (1) exhibit differential expression across the samples; (2) be present in the global human transcription network; and (3) not have more than 10 regulators as parents in the global human transcription network. The first criterion was satisfied by selecting the top 500 genes with the largest variation as measured by the magnitude of the standard deviation of the expression values across samples. The second and third criteria were then applied to this list of 500 genes to

identify genes in the network with 10 or fewer regulators. We note that while it is possible that a gene with more than 10 regulators could mechanistically participate in a strong regulatory relationship, this relationship will not be identifiable with a small dataset in a multinomial model such as we are using here. In a multinomial model, the number of parameters increases exponentially with the number of regulators, making any relationship in a highly connected gene weak. As such, by eliminating genes with more than 10 regulators we are eliminating genes that are unlikely to score well.

Data discretization

The scoring metric used by RegNetB requires that the data be discretized. The data for this study were binned into three states describing a high, medium and low expression level for the variables. The bin sizes were evenly distributed across samples for each variable generating a discretized data set in which variables have their top 1/3 of the data entries based on expression as "high", the bottom 1/3 of the entries as "low" and the remainder 1/3 of the entries as "medium". This binning strategy has been used elsewhere and has been shown empirically to be robust in capturing relevant details of the systems under study [36, 40, 48, 49].

IV.3 Results and Discussion

Global human transcription factor-gene network

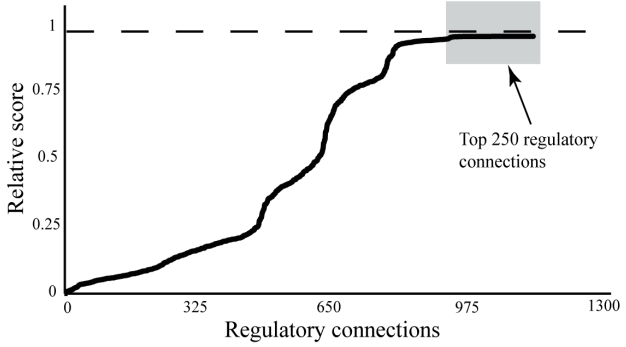
The final global human transcription factor-gene bipartite network generated from the MsigDB consists of 12,026 gene symbols and 392 regulators with a total of 134,874

regulator-gene associations. From these 392 regulators or regulatory regions, 217 were associated with known transcription factors names. The remaining 175 regulators (UK1 to UK175) consisted of 60 known regulatory sequences documented in TRANSFAC and 115 regulatory sequences found and documented elsewhere [9]. After filtering, we compiled a final list of 253 genes and 292 regulators interconnected in a bipartite network with 1,266 connections.

Strongest connections identified by RegNetB

Figure 11(A) shows the score distribution of transcription factor-gene connections. Based on this distribution, we selected the connections that ranked at the top area of the curve illustrated in the Figure 11(A). This group of regulatory connections shows a clear similarity in terms of the regulatory strength. A total of 250 regulatory connections were collected, all with a score >0.993. Figure 11(B) shows the top 10 connections from this list.

(A)



(B) **Top 10 connections**

Regulator	Gene
PAX4	RLN1
TCF3	EPCAM
UK15	RLN1
SP1	SEC13
PAX4	RLN2
SRF	MYL9
SP1	ATP6V0E1
SRF	CSRP1
JUN	ACPP
UK15	RLN2

Figure 11. Connections ranked by score: (A) Relative score distribution for the regulatory connections kept after RegNetB analysis. The shadowed region shows the top 250 connections based on score. (B) Top 10 connections predicted by RegNetB.

We noticed that not all connections associated with a regulator included in the top 250 strongest connections list were part of the group of top connections. This relative strength distribution implies that some regulatory connections associated with a specific regulator play a more relevant regulatory function than the others.

PAX4 regulatory role

Regulation of RLN1 by PAX4 ranked top on the list of strong connections in Figure 11(B). Similarly, the regulation of RLN2 by PAX4 also ranked well (fifth position). RLN1 and RLN2 have been associated with prostate cancer in other studies [60]. The regulator PAX4 has been identified as a tumor suppressor in melanoma studies [61], however has not been associated with prostate cancer [62].

To further evaluate the RegNetB prediction of PAX4's influence on RLN1 and RLN2, we examined the expression levels of the target genes and any other regulator(s) associated with the genes. As shown in Figure 12(A), the expression patterns of RLN1 and RLN2 share a strong similarity in terms of regulation not only by the topological model but also by the coordinated linear pattern observed in the data. This observation supports the prediction that PAX4 is a common factor responsible for changes in the expression of RLN1 and RLN2.

ACPP (PAP) regulation

Another connection observed in the strong connection list shown in Figure 11(B) was the regulation of ACPP by JUN. ACPP or PAP (Prostatic Acid Phosphatase) is a known prostate cancer marker used to monitor tumor progression and/or patients improvement [63]. RegNetB suggested that the main regulatory activity associated with

this gene is best described by the regulators JUN, BACH1, and BACH2. JUN is an oncogene that has been associated with different types of cancer including prostate cancer tumor progression [64, 65]. In the case of BACH1 and BACH2, even though there are some associations with breast cancer and leukemia[66, 67], we found no links with prostate tumor progression.

MAZ and TAF co-regulation

To further explore RegNetB's results, we examined sets of two or more genes that shared the same group of regulators within the selected list of 250 regulatory connections. We found two genes, PGC and GDF15 that are both co-regulated by TAF and MAZ. Both PGC and GDF15 have been associated with prostate cancer and have been documented as potential biomarkers [68-70]. Figure 12(B) shows coordinated patterns between these genes but not in a linear manner. Interestingly, MAZ and TAF have been associated with other types of cancer [71-73], but we found no reports associating MAZ and TAF with prostate cancer.

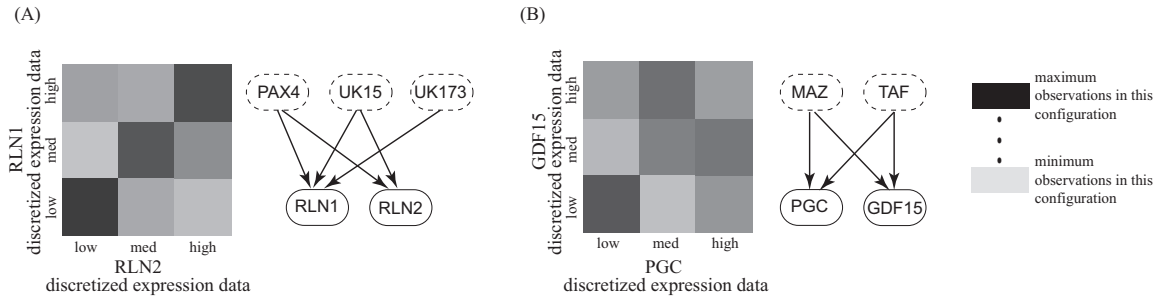


Figure 12. Top scoring regulatory relationships and discretized data patterns. Each grid in (A) and (B) shows the nine possible state combinations in which each pair of variables is observed in the discretized expression data. In the regulatory networks, the dotted ovals represent regulators while the solid ovals represent target genes. (A) RLN1 and RLN2 expression and regulatory network. Note that RLN1 and RLN2 show a nearly linear co-expression pattern. (B) PGC and GDF15 expression and regulatory network. The expression pattern of PGC relative to GDF15 does not show a linear pattern, but still scores well in the multinomial model used by RegNetB.

These results suggest that RegNetB is able to identify physiologically relevant regulatory protien-gene relationships based on gene expression data. Many of the target genes identified by RegNetB have been implicated in prostate cancer progression, but the relevant regulatorion is largely new. In particular, RegNetB identified the regulators PAX4, BACH1, BACH2, MAZ and TAF as playing a central role in this prostate cancer gene expression data set.

The method used by RegNetB can be directly applied to any gene expression dataset, as long as a transcriptional regulatory network is known for the organism. By identifying explanatory regulatory protien-gene relationships, RegNetB allows a researcher to look beyond changes in gene expression, and start to identify possible causes for that change in expression.

CHAPTER V

Conclusions

In this thesis I showed how integration of gene expression data with existing knowledge about gene networks can identify parts of the gene network that are relevant to the specific cellular conditions contained in the data. My study focused on transcriptional regulation, but a similar approach can be applied to other biological networks. A key feature of the methods I have developed in this thesis is the ability to infer relationships in observed variables through a network containing both observed and unobserved elements.

Understanding how genes are regulated is an important step to identify a molecular mechanism associated with a condition. Furthermore, if this condition is a disease, the understanding of key genes and proteins relationships provides new research opportunities to develop new treatments or drug designs.

In the following three sections I summarize the major findings and conclusions on each of the research chapters II, III and IV. I also include a fourth section with new research directions motivated by this work.

V.1 Regulatory relationships identification with unobserved regulators

In Chapter II I addressed how to identify most explanatory regulator-gene relationships using only observed gene expression data. Using synthetic data and a small network I showed that the original network could be recovered reliably using gene expression data alone. The relationships captured in the network included both linear and non-linear cases, and the synthetic analysis was done in the presence of significant noise. Interestingly, the simulations showed that after some point, more data do not necessarily mean more accurate results. On the other hand, even when a relatively small dataset, important features can still be obtained.

After evaluating different sets of 2 regulators and 4 genes examples using *E. coli* gene expression data it was clear that when the tested regulatory associations contained genes with a strong signal, the predicted true network scored significantly better than any other network containing the same genes. Also, when the predicted models were compared with global regulatory models from literature, the predicted method showed simpler relationships suggesting that literature models contained relationships that were not supported by the dataset used.

The ability of this method to identify specific and true regulatory relationships even with the limitation of having hidden variables motivated further studies in this area to improve the efficiency of the method in terms of scalability and computational speed.

V.2 Integration of gene expression data and general transcriptional regulation knowledge

Chapter III introduced a novel approach, POBN2, to systematically integrate transcriptional regulatory networks with gene expression data to identify inactive or false connections. The main advantage of POBN2 is that it can be applied to large networks.

POBN2 addresses the main limitation found on traditional sequence analysis methods: the identification of false predictions. The reduced network suggested by POBN2 provides a more clear and accurate view of the regulatory associations that are both mechanistically sound and maximally consistent with the expression data. POBN2 can also be used to identify active signaling pathways or portions of a known pathway that is highly active if experimental data and some knowledge of the underlying topology are known.

V.3 Identification of the most important regulator-gene activities associated with a disease

In Chapter IV I changed focus from non-relevant or false connections to connections that were more active in the network based on the data. After the analysis of results in Chapter III I observed that the list of the strongest connections based on score did not significantly change during the optimization of the initial network. This observation not only suggested that these connections consistently scored well on each independent optimization steps but that these group of connections play an important role explaining the genes expression variances observed in the dataset.

The human prostate cancer dataset studied in this chapter revealed a group of strong connections in a human transcription network that are more likely to be associated with prostate tumor progression or prostate cells malfunctioning that could promote localized prostate tumors. Although most of the genes included in the top regulator-gene associations group are genes that have been associated in the past with prostate tumor progression based on literature review, some of the regulators had not been implicated and neither their regulatory roles associated with the disease.

These computational approaches developed in this work (POBN2 and RegNetB) to integrate expression data and transcription networks allow a researcher to look beyond a group of differentially expressed genes between two conditions and identify possible regulatory mechanisms that cause the expression changes.

V.4 Future research

In the next sections I include some ideas for future research mainly motivated by some of the challenges observed in this work.

V.4.1 New sampling approaches

One of the limitations of working with hidden variables is the computational time associated with sampling across hidden variables. For methods similar to the one presented in this work, some exploration can still be done to improve the sampling computational time without sacrificing consistency and accuracy on the predictions.

For the system studied in Chapter IV, all regulators in the global network were usually present when the network was mapped against different list sizes of the top differentially expressed genes based on variance. Further selection of genes and the graph portion associated with these top differentially expressed genes were also filtered based on the number of parents. Genes with many parents can be difficult to analyze not only because the signal in the data could not be enough for parameters estimation and connections score evaluation but also because it can represent a challenge in terms of memory demand.

One possible approach to test after following the method presented in Chapter III and IV is to vary the number of highly connected genes to the network. Before the sampling process, genes having a maximum of X parents = 3, 4, 5, 8, 10 can be selected as a starting point (one x_i at a time). For each value of x_i, the initial network is sampled to store regulators states configurations. After this sampling, genes having $x_j > x_i$ parents can be added to the sampled network. Each connection can be scored using the sampled regulators state configurations values obtained with x_i parents even when some additional connections were added to the sampled network.

Three key questions could be answered from this analysis. First, are the predictions for the genes consistent when sampled values for the regulators are fixed based on a smaller portion of the network (x_i vs. x_j)? Second, how sensitive are the predictions to the difference between x_i and x_j? Third, can a similar approach be used to predict missing connections on the initial network (possible false negatives in the initial network?

V.4.2 Regulatory networks learned from data alone

In Chapter II we showed that it is possible to accurately learn a network with hidden regulators using only data from the observed regulated variables. The challenge here was again the sampling method to score each possible network. One interesting feature of the networks learned using only the data from the observed variables is that the regulators not only can represent proteins but any chemical or environmental factor that might affect the expression of the observed variables. In that sense, a network learned directly from data and in which the regulators are modeled as hidden variables will contain a more complete view of the factors regulating the genes.

Alternate sampling approaches can be explored to overcome the computational time limitation. Methods such as variational bayesian learning [53] have been suggested as a computationally faster alternative for hidden variable states sampling. However, the optimization or scoring process when using variational Bayes methods involves the assumption of an auxiliary distribution over the hidden variables [74]. Two main questions can lead this research: First, is there a systematic way that can be develop using the observed variables information in a system to suggest a sounded probability distribution for the hidden variables? Second, for a specific type of regulatory networks, e.g. transcriptional regulatory networks, is there a typical or recurrent probability distribution that can be assumed as a general approximation?

Furthermore, even with a faster sampling method, some areas of the network learning steps need to be explored. For example, how many hidden regulators are optimums to construct such network? As a first approximation, information from

transcriptional networks and pathways associated with the genes to be analyzed can be used to estimate a total number of starting regulators in the network. In addition, some initial structural parts of the network can be fixed or suggested based on clustering or mutual information analysis on the observed variables prior to the network learning steps.

Another area to explore in this network learning problem using only expression data is how this type of more global regulatory networks compares with other networks such as pathways, transcriptional networks, etc.? Can the identity of some of the hidden regulators be deduced, for example the proteins suggested as regulators in pathways and transcriptional networks, by direct comparison of the networks structures?

V.4.3 Tissue specific regulatory associations and causality

For a specific biological condition, e.g. a specific cell type, the approaches presented in chapter III and IV can be used to identify the most active regulatory associations describing the difference in expression of the genes. These regulatory activities may include common regulatory steps needed by most cells as well as cell or tissue specific regulatory activities. By comparing independent networks learned from two different cells/tissues/conditions, can the normal regulatory associations common for most cells versus the ones that are tissue/cells or condition specific be differentiated?

Furthermore, can a disease condition or gene profile describing a disease be compared to what it is expected to be a normal gene profile to identify a small causal group of genes associated with the transition of the normal state to the disease state? Using a network learned from normal tissue samples, identification of a small subset of

key variables that could be the cause of an abnormal tissue condition can be suggested. Most of the differences in the expression of genes arise from changes in expression of a few causal variables. Sampling methods can be designed to predict the top candidate causal variables responsible of the global differences between two tissue profiles. Using a learned network and a distance metric, the difference between two profiles can be quantified after a single perturbation is suggested in a normal profile.

Hidden variables analysis in general is a challenging problem. Nevertheless, the network-based models used in computational biology can benefit from the expressiveness and representational accuracy of models containing hidden variables. The methods presented here provide a practical alternative to computationally expensive hidden variable methods currently available. Furthermore, the methods developed in this thesis provide the foundation to further explore interesting areas such as regulatory networks and biological pathways with more flexibility and in more depth.

Appendix 1

POBN2 single round source code

```
#       Created by: Angel Alvarez
#       angelpr@umich.edu
#       Date: 09-21-09
###################################
# This is a python code that uses PEBL, a python library, (see reference below) to sample from a
# bipartite network with hidden variables and gene expression data to calculate the BDe score of the
# initial network and any resultant network after disconnecting one edge at a time.
# Use a prior graph with hidden regulators containing active and inactive connections to
# approximate missing entries states configuration to identify most of the inactive edges.
# Only present connections in this prior graph will be evaluated to identify the weakest and
# strongest connections.
# This script run a single POBN2 optimization round. For subsequent analysis, data and graph
# need to be updated based on each POBN2 optimization rounds results.
# PEBL needs to be installed to run this script. PEBL downloads can be found at
# http://code.google.com/p/pebl-project/ or more specific Installation instructions can be
# found at http://ano.malo.us/pebl/docs/

import datetime
from pebl import cpd                                    # PEBL need to be installed to import this module
cpd.MultinomialCPD = cpd.MultinomialCPD_Py              # When working with many hidden variables
                                                        # this is needed to avoid memory problems.
from pebl import data, network, evaluator               # normal modules from PEBL to handle data,
                                                        # networks and scoring processes.
import numpy as N                                       # For the data generation/modification process
from pebl.util import unzip                             # For the data modification process (when
                                                        # substituting missing entries for sampled values.

import cPickle
import numpy
from numpy import *
import random

# General variables:
WINDOW=250                                              # Number of missing entries state configurations to
                                                        # be used for averaging networks scores
DATAFILE1="final_contXX_NAT_data_set_r4.txt"            # File containing gene expression and list of
                                                        # regulators based on starting graph used.
                                                        # All entries for regulators should be an 'X'
DATAFILE2="final_cont_NAT_data_set_dummy_r4.txt"        # Same data with dummy entries for
                                                        # regulators. Dummy values will eventually
                                                        # be substituted by the sampled values after
                                                        # Gibbs sampling. It is just to create a data
```

```
                                                    # object with no missing entries.
PRIOR_NET="final_prior_graph_NAT_data_set_r4.dat"   # Initial global graph as a tupple list of
                                                    # connections
OUTFILE1="scoring_edges_effect_NAT_set_r4.txt"      # computational time monitoring data
OUTFILE2="connections_results_NAT_set_r4.dat"       # Dictionary with results. Evaluated
                                                    # connections are the keys
OUTFILE3="results_r4.txt"                           # results as a tab delimitted txt file.

# Reading network and data files
d=data.fromfile(DATAFILE1)
d2=data.fromfile(DATAFILE2)
# Discretizing data into three equal bin size (3 states). If data is discretized, these
# steps are not needed.
includevars = [i for i in range(len(d.variables)) if not all(d.missing[:,i])]
d.discretize(includevars=includevars, numbins=3)
hiddenvars=[i for i in range(len(d.variables)) if all(d.missing[:,i])]
for i in hiddenvars:
        d.variables[i].arity=3
d2.discretize(numbins=3)
# End of discretization. Start reading graph and identifying missing entries indexes.
f=open(PRIOR_NET,"r")
edges_list=cPickle.load(f)
f.close()
net=network.fromdata(d)
net.edges.clear()
net.edges.add_many(edges_list)
missing_indices = unzip(N.where(d.missing==True))
print ""
print "graph and prior network read succesfully"
print ""

# Gibbs evaluator, scoring prior graph and missing values retrieval:
        # Gibbs sampling burn in will run for about 1 hour (these samples will be discarded):
        # PEBL provides to set the burn in in terms of number of iterations. Default is 10 iterations
        # One iteration is one round of sampling for all variables.
burn_time=0
now=datetime.datetime.now()
print "One Gibbs sampling iteration started at "+str(datetime.datetime.now())
s1=""
s1+="One iteration started:\t"+str(datetime.datetime.now())+"\n"
sc1=evaluator.MissingDataNetworkEvaluator(d,net, max_iterations="1")
sc1.score_network()
gstates1=sc1.gibbs_state
now_end=datetime.datetime.now()
print "One Gibbs sampling iteration ended at "+str(datetime.datetime.now())
print ""
s1+="One iteration ended:\t"+str(datetime.datetime.now())+"\n"
while burn_time <= 1 and burn_time >=0:
        sc1=evaluator.MissingDataNetworkEvaluator(d,net, max_iterations="1", gibbs_state=gstates1)
        scored_prior=sc1.score_network()
        gstates1=sc1.gibbs_state
        now2=datetime.datetime.now()
        burn_time=now2.hour-now.hour
s1+="burn in ended:\t"+str(datetime.datetime.now())+"\n"
f21=open("burn_in_time.txt","w")
f21.write(s1)
```

```
f21.close()
print "scored prior after burn in : "+str(scored_prior)
print ""
print "Starting creating and storing Window values for missing entries at "+str(datetime.datetime.now())
s1+="Start window:\t"+str(datetime.datetime.now())+"\n"
# Scoring prior avg over WINDOW values:
results=[]                              # all initial global graph scores: one for each collected set.
gibbs_list=[]                           # all gibbs values within WINDOW used to avg-score each net
for x in range(WINDOW):
        sc1=evaluator.MissingDataNetworkEvaluator(d,net, max_iterations="1", gibbs_state=gstates1)
        results.append(sc1.score_network())
        gstates1=sc1.gibbs_state
        gibbs_list.append(sc1.gibbs_state.assignedvals)          # "WINDOW" set of configuration values
print "Ended creating and storing Window values for missing entries at "+str(datetime.datetime.now())
print ""
s1+="End window:\t"+str(datetime.datetime.now())+"\n"
f4=open("gibbs_list_window.dat","w")                             # Storing missing entries configuration set
cPickle.dump(gibbs_list, f4)
f4.close()
av=0.0
for z in results:
        av+=z
av=av/len(results)
results.append(av)
print "Ended scoring prior using window values at "+str(datetime.datetime.now())
print ""
# Disconnecting each edge (one at a time) and scoring network with WINDOWS gibbs sampled values for
# missing entries to estimate each edge effect on global initial network.
print "started scoring networks with WINDOW values after disconnecting edges:
"+str(datetime.datetime.now())
s1+="Start scoring nets:\t"+str(datetime.datetime.now())+"\n"
results2={}
for y in gibbs_list:
        d2.observations[unzip(missing_indices)] = y
        sc2=evaluator.SmartNetworkEvaluator(d2,net)
        if "PRIOR" in results2.keys():
          results2["PRIOR"].append(sc2.score_network())
        else:
          results2["PRIOR"]=[sc2.score_network()]
        for x in edges_list:
          sc2.alter_network(remove=[x])
          if x in results2.keys():
                  results2[x].append(sc2.score_network())
                  sc2.alter_network(add=[x])
          else:
                  results2[x]=[sc2.score_network()]
                  sc2.alter_network(add=[x])
print "ended scoring networks with WINDOW values after disconnecting edges:
"+str(datetime.datetime.now())
print ""
print "started calculating networks avg scores: "+str(datetime.datetime.now())
final_results={}
for x in results2:
        scores=results2[x]
        avg_score=0.0
        for z in scores:
```

```
            avg_score+=z
        avg_score=avg_score/len(scores)
        final_results[x]=avg_score
print "ended calculating networks avg scores: "+str(datetime.datetime.now())
s1+="End scoring nets:\t"+str(datetime.datetime.now())+"\n"
f6=open(OUTFILE1,"w")
f6.write(s1)
f6.close()
f7=open(OUTFILE2, "w")
cPickle.dump(final_results, f7)
f7.close()
s1=""
for x in final_results:
        s1+=str(x)+"\t"+str(final_results[x])+"\n"
f8=open(OUTFILE3,"w")
f8.write(s1)
f8.close()
```

Appendix 2

Synthetic data generator

```
# 01-19-2010 BY: Angel Alvarez
# This is a python script that generates synthetic discrete gene expression data for the bipartite network
# defined in “edges_list”. The data is generated based on a conditional probability table specified in
# the script for the network (“edges_list”). Variables are allowed a maximum of 3 states (0,1,2).
# The format of the final data file stored in OUTFILE1 will have the required format to be used by PEBL,
# a python library for Bayesian networks learning. PEBL downloads can be found at
# http://code.google.com/p/pebl-project/ or more specific Installation instructions can be
# found at http://ano.malo.us/pebl/docs/

import random
import cPickle

# General variables and constants:
VARS=13                         # Total variables in the study
CUE=4                           # cue nodes (parents)
SAMPLES=5000                    # size of the data set
OUTFILE1="data_file1.txt"
# The graph used as the initial graph is defined as a tupple of connections. The first integer
# in each tuple correspond to a regulator or parent node and the second integer in the tuple is
# a child node or regulated/observed variable. In this case, the four regulators are 0-3 and
# the regulated observed variables are 4-12.
edges_list=[(0,4),(0,5),(0,6),(0,7),(1,7),(1,8),(1,9),(2,7),(3,10),(3,11),(3,12)]
nodesList=range(VARS)
parents=range(CUE)
children=nodesList[CUE:]

# function to get a list of parents for a "node" based on a "graph" containing
# connections as a tuple list:
def find_parents(node, graph):
        parent_list=[]
        for x in graph:
          if (node == x[1]) and (x[0] not in parent_list):
                parent_list.append(x[0])
        parent_list.sort()
        return parent_list

# establish the CPTs as dictionaries and defining each var dictionary as a dictionary:
# each cpt[node #]={}. Keys for each var dic (node) are parents state configs, and the dict
# content are parameters. Example: cpt[5][(0,0,1)]=[0.6,0.4,0.1] means that for node 5,
# having 3 parents in configuration (0,0,1), the parameters are P(node=0)=0.6, P(node=1)=0.4, etc.
cpt={}
for i in range(VARS):
        cpt[i]={}
```

```
# cpt VALUES FOR PARENTS. For 3 states, the greater amount of variations for a parent is when their
# parameters are very similar for any state (1/3 or ~0.33)
for x in parents:
        v1=random.uniform(0.31,0.35)
        v2=random.uniform(0.31,0.35)
        v3=1.0-v2-v1
        cpt[x][()]=[v1,v2,v3]

# CHILD NODES CPT (conditional probability table):
cpt[4][(0,)]=[0.1,0.8,0.1]
cpt[4][(1,)]=[0.2,0.1,0.7]
cpt[4][(2,)]=[0.8,0.1,0.1]
cpt[5][(0,)]=[0.2,0.6,0.2]
cpt[5][(1,)]=[0.1,0.2,0.7]
cpt[5][(2,)]=[0.7,0.1,0.2]
cpt[6][(0,)]=[0.1,0.7,0.2]
cpt[6][(1,)]=[0.1,0.1,0.8]
cpt[6][(2,)]=[0.6,0.2,0.2]
cpt[7][(0,0,0)]=[0.7,0.2,0.1]
cpt[7][(0,0,1)]=[0.6,0.2,0.2]
cpt[7][(0,0,2)]=[0.8,0.1,0.1]
cpt[7][(0,1,0)]=[0.7,0.2,0.1]
cpt[7][(0,1,1)]=[0.6,0.2,0.2]
cpt[7][(0,1,2)]=[0.8,0.1,0.1]
cpt[7][(0,2,0)]=[0.6,0.1,0.3]
cpt[7][(0,2,1)]=[0.8,0.1,0.1]
cpt[7][(0,2,2)]=[0.2,0.6,0.2]
cpt[7][(1,0,0)]=[0.7,0.2,0.1]
cpt[7][(1,0,1)]=[0.6,0.2,0.2]
cpt[7][(1,0,2)]=[0.7,0.2,0.1]
cpt[7][(1,1,0)]=[0.7,0.2,0.1]
cpt[7][(1,1,1)]=[0.1,0.8,0.1]
cpt[7][(1,1,2)]=[0.2,0.6,0.2]
cpt[7][(1,2,0)]=[0.7,0.1,0.2]
cpt[7][(1,2,1)]=[0.2,0.2,0.6]
cpt[7][(1,2,2)]=[0.1,0.1,0.8]
cpt[7][(2,0,0)]=[0.7,0.2,0.1]
cpt[7][(2,0,1)]=[0.6,0.2,0.2]
cpt[7][(2,0,2)]=[0.7,0.1,0.2]
cpt[7][(2,1,0)]=[0.8,0.1,0.1]
cpt[7][(2,1,1)]=[0.1,0.8,0.1]
cpt[7][(2,1,2)]=[0.2,0.2,0.6]
cpt[7][(2,2,0)]=[0.1,0.7,0.2]
cpt[7][(2,2,1)]=[0.2,0.2,0.6]
cpt[7][(2,2,2)]=[0.1,0.1,0.8]
cpt[8][(0,)]=[0.7,0.2,0.1]
cpt[8][(1,)]=[0.1,0.8,0.1]
cpt[8][(2,)]=[0.2,0.1,0.7]
cpt[9][(0,)]=[0.8,0.1,0.1]
cpt[9][(1,)]=[0.1,0.7,0.2]
cpt[9][(2,)]=[0.1,0.1,0.8]
cpt[10][(0,)]=[0.8,0.1,0.1]
cpt[10][(1,)]=[0.1,0.8,0.1]
cpt[10][(2,)]=[0.1,0.1,0.8]
cpt[11][(0,)]=[0.6,0.2,0.2]
```

```
cpt[11][(1,)]=[0.1,0.6,0.3]
cpt[11][(2,)]=[0.1,0.1,0.8]
cpt[12][(0,)]=[0.1,0.2,0.7]
cpt[12][(1,)]=[0.1,0.7,0.2]
cpt[12][(2,)]=[0.7,0.2,0.1]

# sampler function: provide list [v1, v2, v3...] where v1,v2,v3 are parameters (probability values)
# and get out a state
def sampler(probs):
        r=random.random()
        sum=0.0
        count=0
        for p in probs:
          sum+=p
          if sum>r:
                return count
          count+=1
        return len(probs)-1

# Generating data:
mydata=[]
state={}
for x in range(SAMPLES):
        sample_values=[]
        for a in parents:
          state[a]=sampler(cpt[a][()])
          sample_values.append(state[a])
        for b in children:
          if len(find_parents(b,edges_list)) == 0:        # orphans
                sample_values.append(sampler(cpt[b][()]))
          else:
                pstate=[]                                       # To store parents state on sample
                node_parents=find_parents(b,edges_list)     # finding child parent(s)
                for p in node_parents:
                        pstate.append(state[p])
                cptvals=cpt[b][tuple(pstate)]
                sample_values.append(sampler(cptvals))
        mydata+=[sample_values]

# saving data as a string/txt ready to be used with PEBL
s=''
for x in nodesList:
        if x == nodesList[len(nodesList)-1]:
          s+=str(x)+",discrete(3)"+"\n"         # This heading will tell PEBL the name of the variable
        else:                                   # (here are integers), that the variable is
          s+=str(x)+",discrete(3)"+"\t"         # "discrete" and it has 3 states.
for x in mydata:
        for y in range(len(x)):
          if y == len(x)-1:
                s+=str(x[y])+"\n"
          else:
                s+=str(x[y])+"\t"
r1=open(OUTFILE1,"w")
r1.write(s)
r1.close
```

BIBLIOGRAPHY

1. Brown, T., *Analysis of RNA by northern and slot-blot hybridization.* Curr Protoc Immunol, 2001. **Chapter 10**: p. Unit 10 12.
2. VanGuilder, H.D., K.E. Vrana, and W.M. Freeman, *Twenty-five years of quantitative PCR for gene expression analysis.* Biotechniques, 2008. **44**(5): p. 619-26.
3. Subramanian, A., et al., *Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles.* Proc Natl Acad Sci U S A, 2005. **102**(43): p. 15545-50.
4. Blais, A. and B.D. Dynlacht, *Constructing transcriptional regulatory networks.* Genes Dev, 2005. **19**(13): p. 1499-511.
5. Cecchini, K.R., A.R. Banerjee, and T.H. kim, *Towards a genome-wide resconstruction of cis-regulatory networks in the human genome.* Seminars in Cell & Developmental Biology, 2009. **20**(7): p. 7.
6. Sabatti, C. and G.M. James, *Bayesian sparse hidden components analysis for transcription regulation networks.* Bioinformatics, 2006. **22**(6): p. 739-46.
7. Matys, V., et al., *TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes.* Nucleic Acids Res, 2006. **34**(Database issue): p. D108-10.
8. Grabe, N., *AliBaba2: context specific identification of transcription factor binding sites.* In Silico Biol, 2002. **2**(1): p. S1-15.
9. Xie, X., et al., *Systematic discovery of regulatory motifs in human promoters and 3′ UTRs by comparison of several mammals.* Nature, 2005. **434**(7031): p. 338-45.
10. Abruzzese, F., et al., *Lack of correlation between mRNA expression and enzymatic activity of the aspartate aminotransferase isoenzymes in various tissues of the rat.* FEBS Lett, 1995. **366**(2-3): p. 170-2.
11. Kirkpatrick, K.L., R.F. Newbold, and K. Mokbel, *There is no correlation between c-Myc mRNA expression and telomerase activity in human breast cancer.* Int Semin Surg Oncol, 2004. **1**(1): p. 2.
12. Gama-Castro, S., et al., *RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation.* Nucleic Acids Res, 2008. **36**(Database issue): p. D120-4.
13. Shachter, R.D., *Evaluating Influence Diagrams.* OPERATIONS RESEARCH, 1986. **34**(6): p. 871-882.
14. Shachter, R.D., *Probabilistic inference and influence diagrams.* OPERATIONS RESEARCH, 1988. **36**(4): p. 589-605.
15. Shafer, G.R. and P.P. Shenoy, *Probability propagation.* Annals of Mathematics and Artificial Intelligence, 1990. **2**(1): p. 327-351.

16. Neapolitan, R.E., *Learning Bayesian Networks*. 2003: Prentice-Hall, Inc.
17. Cooper, G.F. and E. Herskovits, *A Bayesian method for the induction of probabilistic networks from data.* Machine Learning, 1992. **9**(4): p. 39.
18. Heckerman, D., *A tutorial on learning with Bayesian networks*, in *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*. 1998, Kluwer Academic Publishers: Erice, Italy. p. 301-354.
19. Chickering, D.M., D. Geiger, and D. Heckerman, *Learning Bayesian Networks is NP-Hard.* Microsoft Research, 1994. **22**.
20. Heckerman, D., *Learning in Graphical Models.* MIT Press, Cambridge, MA, 1999.
21. Ghahramani, Z., *An introduction to hidden Markov models and Bayesian networks*, in *Hidden Markov models: applications in computer vision*. 2002, World Scientific Publishing Co., Inc. p. 9-42.
22. Gilks, W.R., *Markov Chain Monte Carlo in Practice.* 1995.
23. Riggelsen, C., *Learning parameters of Bayesian networks from incomplete data via importance sampling.* International Journal of Approximate Reasoning, 2006. **42**(1-2): p. 15.
24. Bulyk, M.L., *Computational prediction of transcription-factor binding site locations.* Genome Biol, 2003. **5**(1): p. 201.
25. Kel, A.E., et al., *MATCH: A tool for searching transcription factor binding sites in DNA sequences.* Nucleic Acids Res, 2003. **31**(13): p. 3576-9.
26. Ohler, U. and H. Niemann, *Identification and analysis of eukaryotic promoters: recent computational approaches.* Trends Genet, 2001. **17**(2): p. 56-60.
27. Heintzman, N.D. and B. Ren, *The gateway to transcription: identifying, characterizing and understanding promoters in the eukaryotic genome.* Cell Mol Life Sci, 2007. **64**(4): p. 386-400.
28. Jain, A.K., M.N. Murty, and P.J. Flynn, *Data clustering: a review.* ACM Comput. Surv., 1999. **31**(3): p. 264-323.
29. Asyali, M.H., et al., *Gene Expression Profile Classification: A Review.* Current Bioinformatics, 2006. **1**: p. 55-73.
30. Kerr, G., et al., *Techniques for clustering gene expression data.* Comput Biol Med, 2008. **38**(3): p. 283-93.
31. Brynildsen, M.P., et al., *Biological network mapping and source signal deduction.* Bioinformatics, 2007. **23**(14): p. 1783-91.
32. Beal, M.J., et al., *A Bayesian approach to reconstructing genetic regulatory networks with hidden factors.* Bioinformatics, 2005. **21**(3): p. 349-56.
33. Zare, H., et al., *Reconstruction of Escherichia coli transcriptional regulatory networks via regulon-based associations.* BMC Syst Biol, 2009. **3**(1): p. 39.
34. Ernst, J., et al., *A semi-supervised method for predicting transcription factor-gene interactions in Escherichia coli.* PLoS Comput Biol, 2008. **4**(3): p. e1000044.
35. Hartemink, A.J., et al., *Combining location and expression data for principled discovery of genetic regulatory network models.* Pac Symp Biocomput, 2002: p. 437-49.
36. Friedman, N., *Probabilistic models for identifying regulation networks.* Bioinformatics, 2003. **19**(suppl_2): p. ii57-.

37. Beer, M.A. and S. Tavazoie, *Predicting gene expression from sequence.* Cell, 2004. **117**(2): p. 185-98.
38. Woolf, P.J., et al., *Bayesian analysis of signaling networks governing embryonic stem cell fate decisions.* Bioinformatics, 2005. **21**(6): p. 741-53.
39. Shah, A. and P.J. Woolf, *Python Environment for Bayesian Learning: Inferring the Structure of Bayesian Networks from Knowledge and Data.* JMLR, 2009. **10**(Feb): p. 159--162.
40. Margolin, A.A., et al., *ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context.* BMC Bioinformatics, 2006. **7 Suppl 1**: p. S7.
41. Faith, J.J., et al., *Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles.* PLoS Biol, 2007. **5**(1): p. e8.
42. Salmon, K., et al., *Global gene expression profiling in Escherichia coli K12. The effects of oxygen availability and FNR.* J Biol Chem, 2003. **278**(32): p. 29837-55.
43. Blake, T., et al., *Transcription activation by FNR: evidence for a functional activating region 2.* J Bacteriol, 2002. **184**(21): p. 5855-61.
44. Compan, I. and D. Touati, *Anaerobic activation of arcA transcription in Escherichia coli: roles of Fnr and ArcA.* Mol Microbiol, 1994. **11**(5): p. 955-64.
45. Spellman, P.T., et al., *Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization.* Mol Biol Cell, 1998. **9**(12): p. 3273-97.
46. de Lichtenberg, U., et al., *New weakly expressed cell cycle-regulated genes in yeast.* Yeast, 2005. **22**(15): p. 1191-201.
47. Nachman, I., A. Regev, and N. Friedman, *Inferring quantitative models of regulatory networks from expression data.* Bioinformatics, 2004. **20 Suppl 1**: p. i248-56.
48. Yu, J., et al., *Advances to Bayesian network inference for generating causal networks from observational biological data.* Bioinformatics, 2004. **20**(18): p. 3594-603.
49. Friedman, N., *Inferring cellular networks using probabilistic graphical models.* Science, 2004. **303**(5659): p. 799-805.
50. Sachs, K., et al., *Bayesian network approach to cell signaling pathway modeling.* Sci STKE, 2002. **2002**(148): p. pe38.
51. Nilsson, L. and V. Emilsson, *Factor for inversion stimulation-dependent growth rate regulation of individual tRNA species in Escherichia coli.* J Biol Chem, 1994. **269**(13): p. 9460-5.
52. Mallik, P., et al., *Growth phase-dependent regulation and stringent control of fis are conserved processes in enteric bacteria and involve a single promoter (fis P) in Escherichia coli.* J Bacteriol, 2004. **186**(1): p. 122-35.
53. Beal, M.J., Ghahramani, Z., *Variational Bayesian Learning of Directed Graphical Models with Hidden Variables.* Bayesian Analysis, 2006. **1**(4): p. 40.
54. Chandran, U.R., et al., *Gene expression profiles of prostate cancer reveal involvement of multiple molecular pathways in the metastatic process.* BMC Cancer, 2007. **7**: p. 64.

55. Yu, Y.P., et al., *Gene expression alterations in prostate cancer predicting tumor aggression and preceding development of malignancy.* J Clin Oncol, 2004. **22**(14): p. 2790-9.
56. Abate-Shen, C. and M.M. Shen, *Molecular genetics of prostate cancer.* Genes Dev, 2000. **14**(19): p. 2410-34.
57. Shah, A. and P.J. Woolf, *Python Environment for Bayesian Learning: Inferring the Structure of Bayesian Networks from Knowledge and Data.* Journal of Machine Learning Research, 2009. **10**: p. 4.
58. Dai, M., et al., *Web-based GeneChip analysis system for large-scale collaborative projects.* Bioinformatics, 2007. **23**(16): p. 2185-7.
59. Dai, M., et al., *Evolving gene/transcript definitions significantly alter the interpretation of GeneChip data.* Nucleic Acids Res, 2005. **33**(20): p. e175.
60. Feng, S., et al., *Relaxin promotes prostate cancer progression.* Clin Cancer Res, 2007. **13**(6): p. 1695-702.
61. Hata, S., et al., *PAX4 has the potential to function as a tumor suppressor in human melanoma.* Int J Oncol, 2008. **33**(5): p. 1065-71.
62. Robson, E.J., S.J. He, and M.R. Eccles, *A PANorama of PAX genes in cancer and development.* Nat Rev Cancer, 2006. **6**(1): p. 52-62.
63. Shih, W.J., et al., *Serum PSA and PAP measurements discriminating patients with prostate carcinoma from patients with nodular hyperplasia.* J Natl Med Assoc, 1994. **86**(9): p. 667-70.
64. Leaner, V.D., et al., *Inhibition of AP-1 transcriptional activity blocks the migration, invasion, and experimental metastasis of murine osteosarcoma.* Am J Pathol, 2009. **174**(1): p. 265-75.
65. Tiniakos, D.G., et al., *Expression of c-jun oncogene in hyperplastic and carcinomatous human prostate.* Urology, 2006. **67**(1): p. 204-8.
66. Gupta, R., et al., *Analysis of the DNA substrate specificity of the human BACH1 helicase associated with breast cancer.* J Biol Chem, 2005. **280**(27): p. 25450-60.
67. Ono, A., et al., *Nuclear positioning of the BACH2 gene in BCR-ABL positive leukemic cells.* Genes Chromosomes Cancer, 2007. **46**(1): p. 67-74.
68. Antunes, A.A., et al., *The role of prostate specific membrane antigen and pepsinogen C tissue expression as an adjunctive method to prostate cancer diagnosis.* J Urol, 2009. **181**(2): p. 594-600.
69. Vanhara, P., et al., *Growth/differentiation factor-15 inhibits differentiation into osteoclasts--a novel factor involved in control of osteoclast differentiation.* Differentiation, 2009. **78**(4): p. 213-22.
70. Kawahara, T., et al., *Analysis of NSAID-activated gene 1 expression in prostate cancer.* Urol Int, 2010. **84**(2): p. 198-202.
71. Wang, X., et al., *MAZ drives tumor-specific expression of PPAR gamma 1 in breast cancer cells.* Breast Cancer Res Treat, 2008. **111**(1): p. 103-11.
72. Song, J., et al., *Genomic organization and expression of a human gene for Myc-associated zinc finger protein (MAZ).* J Biol Chem, 1998. **273**(32): p. 20603-14.
73. Voulgari, A., et al., *TATA box-binding protein-associated factor 12 is important for RAS-induced transformation properties of colorectal cancer cells.* Mol Cancer Res, 2008. **6**(6): p. 1071-83.

74. Logsdon, B.A., G.E. Hoffman, and J.G. Mezey, *A variational Bayes algorithm for fast and accurate multiple locus genome-wide association analysis.* BMC Bioinformatics, 2010. **11**: p. 58.