12-1-2009

Learning condition-specific networks

Sushmita Roy

SUSHMITA ROY
Candidate

COMPUTER SCIENCE
Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

, Chairperson

Susan R. Atlas

Learning condition-specific networks

by

Sushmita Roy

B.E., University of Pune, 2000
M.S., Computer Science, University of New Mexico, 2005

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Computer Science

The University of New Mexico

Albuquerque, New Mexico

December, 2009

Dedication

To my mother for her strength and inspiration.

Acknowledgments

This dissertation is not complete without its acknowledgements. The list is long, and rightfully so. The work described in this dissertation lies in the intersection of Computer Science and Biology, and would not be possible without two important people: Dr. Terran Lane and Dr. Margaret Werner-Washburne. I am thankful to Dr. Lane for introducing me to machine learning and artificial intelligence, for sharing many lessons, including those about life in general, on how to be a good researcher and a good scientist, and for being a person with inexhaustible patience and encouragement. I am thankful to Dr. Washburne for introducing me to the logic within cells, for teaching me to appreciate all life, unicellular and multicellular, to ask the questions that matter, for her faith in me and for her immensely inspiring positive attitude. I would also like to thank the rest of my PhD committee (Dr. Melanie Moses and Dr. Susan Atlas) for the many exchanges we have had in reading groups, for being interested in my research and for taking the time to be on my committee.

I thank the various funding agencies (National Institute of Health, National Science Foundation, Howard Hughes Medical Institute) for granting us funds to pursue the ideas described in this dissertation. In particular, I would like to acknowledge the grants from NIMH (1R01MH076282-03) and NSF (IIS-0705681) to Dr. Lane, from NIH (GM-67593) and NSF (MCB0734918) to Dr. Washburne and from HHMI-NIH/NIBIB (56005678) to the Program in Interdisciplinary Biological and Biomedical Sciences (PIBBS).

I would like to thank Dr. Bruce Birren for introducing me to Dr. Manolis Kellis, and to Dr. Kellis, for giving me the opportunity to spend a summer in his lab where I learned about comparative genomics. I would like to thank Dr. Alexander Stark for introducing me to the fly model organism and to important bioinformatics resources that are available for fly.

I am thankful to my professors in the UNM Computer science department, from whom I have taken classes, for introducing me to the various facets of computer science research, which are important to obtain a well-rounded perspective of the field of Computer science.

I would like to thank the friendly staff at the Computer science department for helping me with administrative issues and to the Computer science systems group for allowing me to take humongous amounts of disk-space, and for giving me access to CPU resources without which a lot of the experimental work described here would be difficult.

I am thankful to researchers, including those from our lab, who have made their data available for other people to apply their algorithms for novel biological pattern recognition and discovery. I am also thankful to researchers establishing and maintaining databases in a readily downloadable and parsable manner.

I am thankful to past and present members machine learning research group at UNM for sharing and brain-storming papers and discussing important ML concepts, many of which have been important in this work. I am also thankful to the members of the bioinformatics reading group for contributing and brain-storming ideas from biology and computer science. I am also thankful to the past and present members of the Werner-Washburne laboratory for their co-operative and collaborative spirit, and being wonderful colleagues. I am thankful to the PIBBS community at UNM for being such a great group of interdisciplinary researchers.

I am thankful to my friends, Sergey, Eva, and Sahar, for their help and support, for giving me advice and encouragement and listening to my ravings, complaints and frustrations.

I am thankful to my family, my sister, my mother and my father, for their love and faith in me and taking pride in everything thing I have done and telling me from time to time that I am a nerd.

Last, but not the least, I am thankful to Andreas, my friend, philosopher, and soulmate, for his patience, love and support, and for being there for me.

Learning condition-specific networks

by

Sushmita Roy

ABSTRACT OF DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Computer Science

The University of New Mexico

Albuquerque, New Mexico

December, 2009

Learning condition-specific networks

by

Sushmita Roy

B.E., University of Pune, 2000
M.S., Computer Science, University of New Mexico, 2005
PhD., Computer Science, University of New Mexico, 2009

Abstract

Condition-specific cellular networks are networks of genes and proteins that describe functional interactions among genes occurring under different environmental conditions. These networks provide a systems-level view of how the parts-list (genes and proteins) interact within the cell as it functions under changing environmental conditions and can provide insight into mechanisms of stress response, cellular differentiation and disease susceptibility. The principle challenge, however, is that cellular networks remain unknown for most conditions and must be inferred from activity levels of genes (mRNA levels) under different conditions. This dissertation aims to develop computational approaches for inferring, analyzing and validating cellular networks of genes from expression data.

This dissertation first describes an unsupervised machine learning framework for inferring cellular networks using expression data from a single condition. Here cellular networks are represented as undirected probabilistic graphical models and are learned using a novel, data-driven algorithm. Then several approaches are described that can learn networks using data from multiple conditions. These approaches apply to cases where the condition may or may not be known and, therefore, must be inferred as part of the learning problem. For the latter, the condition variable is allowed to influence expression of genes at different levels of granularity: condition variable per gene to a single condition variable for all genes.

Results on simulated data suggest that the algorithm performance depends greatly on the size and number of connected components of the union network of all conditions. These algorithms are also applied to microarray data from two yeast populations, *quiescent* and *non-quiescent*, isolated from glucose starved cultures. Our results suggest that by sharing information across multiple conditions, better networks can be learned for both conditions, with many more biologically meaningful dependencies, than if networks were learned for these conditions independently. In particular, processes that were shared among both cell populations were involved in response to glucose starvation, whereas the processes specific to individual populations captured characteristics unique to each population. These algorithms were also applied for learning networks across multiple species: yeast (*S. cerevisiae*) and fly (*D. melanogaster*). Preliminary analysis suggests that sharing patterns across species is much more complex than across different populations of the same species and basic metabolic processes are shared across the two species.

Finally, this dissertation focuses on validation of cellular networks. This validation framework describes scores for measuring how well network learning algorithms capture higher-order dependencies. This framework also introduces a measure for evaluating the entire inferred network structure based on the extent to which similarly functioning genes are close together on the network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Central to the proper functioning of living systems is the ability to accurately sense environmental cues and respond to changing conditions [48]. This ability of producing different *condition-specific responses* involves global changes at different levels of cellular organization including the transcriptomic, metabolic and proteomic levels. An understanding of condition-specific responses to changing extracellular environments is important to characterize the cellular mechanisms pertaining to growth, maintenance, and cellular differentiation, as well as the failed mechanisms resulting in metabolic and genetic diseases including cancer [78, 124].

Advances in high-throughput technology have led to genome-wide measurements of the activity levels of the cellular parts under different conditions [25, 47], and parts that are differentially expressed are thought to be involved in a condition-specific response [108]. However, the parts lists identified, based only on differential expression, is likely to be incomplete, because there may be genes that individually are not differentially expressed but may be subtly involved in a pathway required for a specific response [30]. More importantly, while differential expression analysis tells us *which* parts are involved, without the knowledge of interactions, we do not know *how* they are involved. To gain a systemic understanding of condition-specific responses, we need to capture condition-specific behavior not as lists of genes but as networks of functional interactions among genes. Condition-specific networks describe functional interactions among genes and other macromolecules under different conditions, providing a systemic view of condition-specific behavior in organisms.

Because we are still technologically limited in our ability to measure the fine-grained interaction patterns among the parts, cellular networks remain unknown for most conditions. Fortunately, statistical machine learning offers us a variety of probabilistic frameworks that reverse engineer the functional interaction network of genes from their activity levels (mRNA) [91, 40, 125, 18, 24, 44, 87]. However, learning condition-specific networks is not a straightforward application of these algorithms, because condition-specific network learning is a multiple network learning problem where networks from different conditions may have varying levels of shared parts. This dissertation identifies the computational and biological challenges in learning condition-specific networks and describes novel machine learning algorithms and their applications for addressing these challenges.

Existing approaches for network-based analysis often infer coarse bi-partite graphs between transcription factors and condition-specific targets [125, 18]. These approaches do not capture the fine-grained interaction patterns among the target genes. Other approaches learn independent networks for different conditions and identify shared and unique parts of the network as a post-processing step [113, 13, 131]. Such approaches do not incorporate the shared information across conditions during network learning and often focus on pairwise co-expression networks. We develop approaches that overcome limitations of existing approaches by (a) simultaneously learning networks from multiple conditions, and (b) learning networks capturing general, higher-order statistical dependencies subsuming co-expression relationships. In particular, we extend the multi-net framework of learning multiple Bayesian networks, one for each class (condition) variable [52], to incorporate shared information during network learning. We also describe several methods based on mixtures of graphs that infer the condition variable during network learning and incorporate sharing at different granularities, from individual genes to the entire network. Our approaches based on the mixture of graphs do not require the condition to be specified as input giving us the flexibility to apply to situations where conditions may or may not be known with certainty. Finally, we describe validation approaches of network inference algorithms for both single as well as multiple networks.

To demonstrate the value of our approach we applied our algorithms to simulated data from networks of known topology, as well as two real-world examples of the condition variable: cell populations and species. Experiments on simulated data gave us a controlled setting to systematically assess the strengths and benefits of our approach and indicated that the topology of the union graph of the condition-specific networks significantly affects performance in addition to the generative model. Application of our algorithms to learn population-specific networks for two yeast stationary-phase cell populations, quiescent and non-quiescent, identified shared processes involved in respiration to be shared across both populations. Application of our algorithms to learn species-specific networks, yeast and fly, showed that basic metabolic processes are conserved across both species. Importantly, population-specific and species-specific networks represent examples at two ends of a spectrum of condition-specific network learning problems and demonstrate the wide applicability of the class of algorithms developed in this work.

To summarize, this dissertation provides a novel, general formulation of learning condition-specific networks that treats the condition variable as an abstract, global variable allowing us to consider existing problems of context-specific learning, including those from biology, as instances of our framework. This formulation of condition-specific network learning makes contributions to both machine learning and biology. On the machine learning side, we describe novel algorithms that allow us to learn multiple networks from noisy biological data that exhibit different amounts and complexities of sharing patterns across conditions. On the biological side, we capture and characterize conserved and unique behavior of populations and species that agree with existing biological knowledge and propose new hypotheses of population and species-specific behaviors that can be validated via future experiments. Our learning framework lays the ground work for exciting future possibilities of learning condition-specific networks across environmental stimuli, cell types, species and diseases that has potential implications in understanding systems-level behavior at the cell, tissue and organism-wide levels of organization.

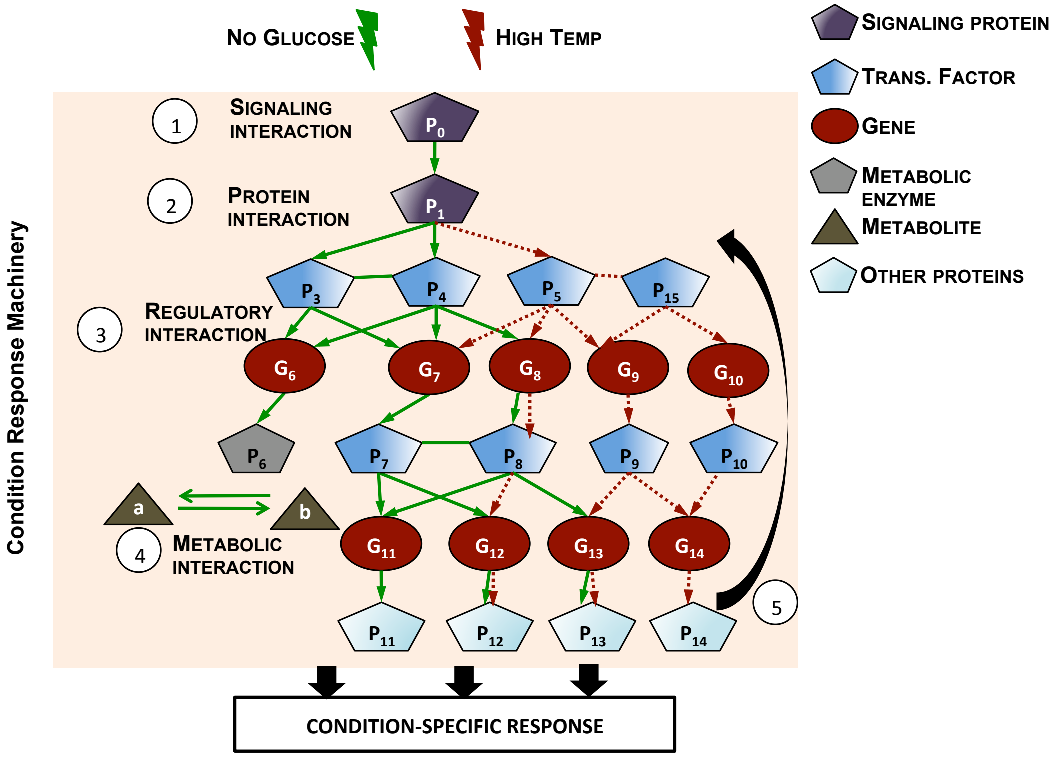


Figure 1.1: Information flow in biological systems for responding to different environmental conditions. Examples are shown of condition variables as stress responses: no glucose and high temperature. The machinery is composed of signaling interactions (1) among signaling proteins that are sensitive to changes in the extra-cellular environment and transmit this information to downstream transcription factors. Transcription factors alone or via protein interactions (2) regulate target genes via regulatory interactions (3). These target genes can code for metabolic enzymes (4) or proteins involved in other condition-specific response function and can ultimately feed back into the system (5).

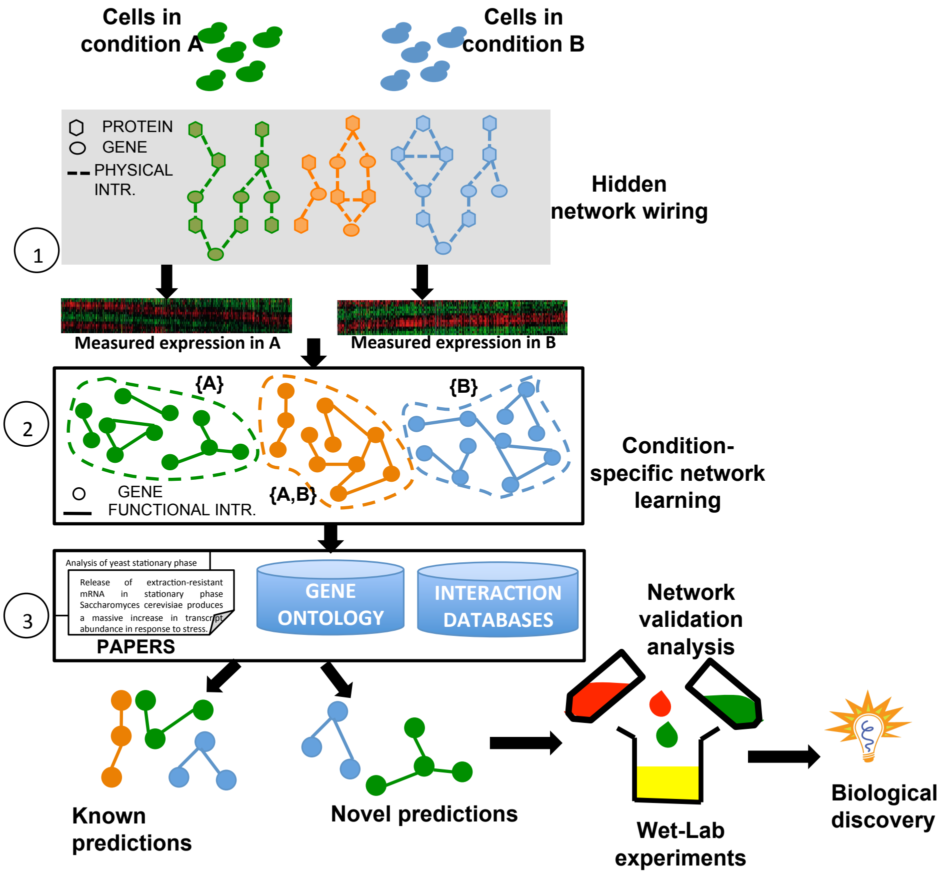


Figure 1.2: Overview of the framework of learning condition-specific networks with the example scenario of two environmental stresses from Fig 1.1. (1)The cell responds to these stresses by global changes in the cellular machinery driven by re-wiring of a hidden physical network, one for each condition as described in Fig 1.1. The different node shapes in the physical network distinguish the gene and protein nodes. The edges of the physical network represent physical interactions (protein-protein and protein-DNA). Although the network is hidden, we can measure activity levels of (some of) the network nodes using microarrays or other high-throughput genome-wide assays. The activity levels of the genes are the outputs of the network re-wiring. (2) The structure of the networks have parts that are shared in both stress networks (orange), and parts that are unique to each network (green or blue). Our goal is to infer functional networks from the observed expression levels, which represent an abstraction of the true hidden physical interactions. (3) After inference, we perform network validation analysis, which validates some of the functional interactions and identifies novel predictions that can be experimentally tested and eventually lead to biological knowledge discovery.

1.1 Overview of the learning condition-specific networks

1.1.1 Information flow in biological systems

Before we proceed with the challenges of learning condition-specific networks, we provide a brief description of the information processing machinery within living cells that is responsible for accurate and timely production of condition-specific responses (Fig 1.1). For ease of exposition we consider two conditions, each representing environmental stresses (No glucose and high temperature) experienced by the unicellular organism, yeast, *S. cerevisiae*. The response to these stresses begins with signaling proteins sensing the changes in the extra-cellular environment and transmitting this information via a cascade of signaling interactions to activate a special class of proteins called transcription factors. The signaling interactions typically involve some form of post-translational modifications such as phosphorylation that change the transcription factors from inactive to active form.

The transcription factors bind to regulatory regions of specific genes to transcriptionally activate the genes. These interactions are called regulatory interactions and bring about the activation of target genes, which code for proteins involved in various functions such as catalyzing a metabolic reaction or downstream transcription factors activating or repressing genes further down the machinery. These proteins can also feed back into the system to maintain the condition-specific response.

The set of physical interactions (signaling[1], protein-protein and regulatory interactions) are all part of the *physical network*. These interactions, which are induced in response to a condition cause a re-wiring of the physical network. The different re-wirings have shared edges producing shared sub-networks that occur in both high temperature or no glucose conditions, as well as edges that occur only in one condition producing the response unique to each condition. Unfortunately the majority of the physical interactions

[1] signaling interaction is a type of protein-protein interaction but is directed

cannot be measured under most conditions making the physical network effectively hidden. However, we can measure the change in expression of some of the network nodes (genes), which can be used to infer a *functional network* where edges represent statistical dependencies. The goal of our condition-specific network learning framework is to infer the functional networks representing condition-specific responses which provides an abstraction of the true wiring of the hidden physical network (Fig 1.2).

1.1.2 Condition-specific network learning framework

Our condition-specific network learning framework defines a *condition* to be a global variable that can represent different environmental stresses, tissues, diseases or even different species. For each value of this condition variable there exists a physical network of cellular parts (genes, proteins and metabolites) that drives genome-wide changes at the transcriptomic, metabolic and proteomic levels. However, we cannot measure the network in each condition itself, but we can obtain genome-wide measurements of the activity levels of genes (proteins and metabolites), which is the output of the network re-wiring. Further, the structure of the networks have parts that are shared in both stress networks (orange edges), and parts that are unique to each network.

We abstract out the physical networks such that the nodes represent the entities (typically genes) that can be measured under each condition and edges representing functional interactions measured by statistical dependencies among the nodes. The problem of condition-specific network learning is defined as inferring the functional interactions among genes using expression measurements of cells under those conditions. After learning the networks, we must analyze them in concert with existing literature from published research, annotation and interaction databases. Some of the sub-networks are expected to be consistent with existing literature (known predictions), while some represent novel relationships among the genes that can be tested in the laboratory leading to biological knowledge discovery.

This dissertation addresses problems in learning condition-specific networks, and validating them using existing knowledge. Further validation and discovery via laboratory experiments is important, but is beyond the scope of this dissertation.

1.2 The computational challenges of learning networks

The problem of learning networks from observed measurements is a well-studied problem in machine learning [42]. The framework for probabilistic graphical models is a widely used framework for representing and learning biological networks because of their natural ability to handle noise and uncertainty [73]. We will review these models in depth in Chapter 2, so we describe them only briefly here. Probabilistic graphical models are composed of two parts: a graph structure describing which nodes interact and a set of functional components, which describe mathematically the nature of dependence. Nodes of the graph represent random variables encoding expression values of genes. However, when we apply this framework to a real-world noisy setting, several questions arise:

Importance of higher-order dependencies Networks in complex domains, including biological systems, have higher-order statistical dependencies – dependencies that occur among more than two random variables [110, 101]. However, mathematical characterization of a higher-order dependency also requires more parameters as opposed to pairwise dependencies, which occur only among two variables. This in turn influences the amount of training data we need to reliably estimate the parameters and the dependencies in the data. In real-world domains, data paucity is common place. This raises the question of how beneficial is it to attempt learning higher-order dependencies when we are limited by data. Are there realistic scenarios where higher-order dependencies are clearly desired over pairwise dependencies? If so, how do we identify such scenarios?

Learning undirected graph structures Biological networks can be represented as directed as well as undirected graphs. We select undirected graphs over directed graphs because we want to identify cyclic dependencies [80], and because of the ambiguity of whether a directed edge in a Bayesian network indicates correlation or causation. Graph structure learning typically requires us to design a score for a candidate graph, and be able to efficiently compute the score for a large number of candidate graphs [65, 63]. Typically in directed graphs, the score is a *decomposable* function of the likelihood of the data given the graph. Decomposability allows the overall score of the graph to be efficiently computed as a summable composition of local graph structures. However, general undirected graphs do not have a decomposable likelihood-based score associated with them [1, 122, 35]. This is because to obtain a valid joint probability distribution we need to compute a partition function, which is NP-hard to compute [1].

Learning multiple networks with shared information We have data from multiple conditions and our goal is to infer a network that describes the condition-specific response. This problem can be formulated in two ways: (a) the condition variable is known (Fig 1.3, Panels (a) and (b)), (b) the condition variable is hidden and must be inferred (Fig 1.3, Panels (c) and (d)). In a machine learning setup, we can use multi-net learning frameworks to address [52] (a) and a mixture of graphs to address (b). However, multi-net frameworks do not share information across conditions, which is limiting in our situation because the networks across conditions are likely to share certain properties among them. We want models that are data-efficient, and can exploit the shared data to learn better, more robust networks. Regarding (b), although mixture of graphs are feasible models in theory, it has been shown in practice only with directed graphs [134] and with decomposable graphs [99]. In contrast, we want to learn mixtures for general undirected graphs. Efficient use of data and parameter sharing can also be achieved by tying the parameters of related probability distributions [125, 41]. Can we learn condition-specific network models that exploit shared information across conditions using either data pooling, mixture models, or parameter tying?

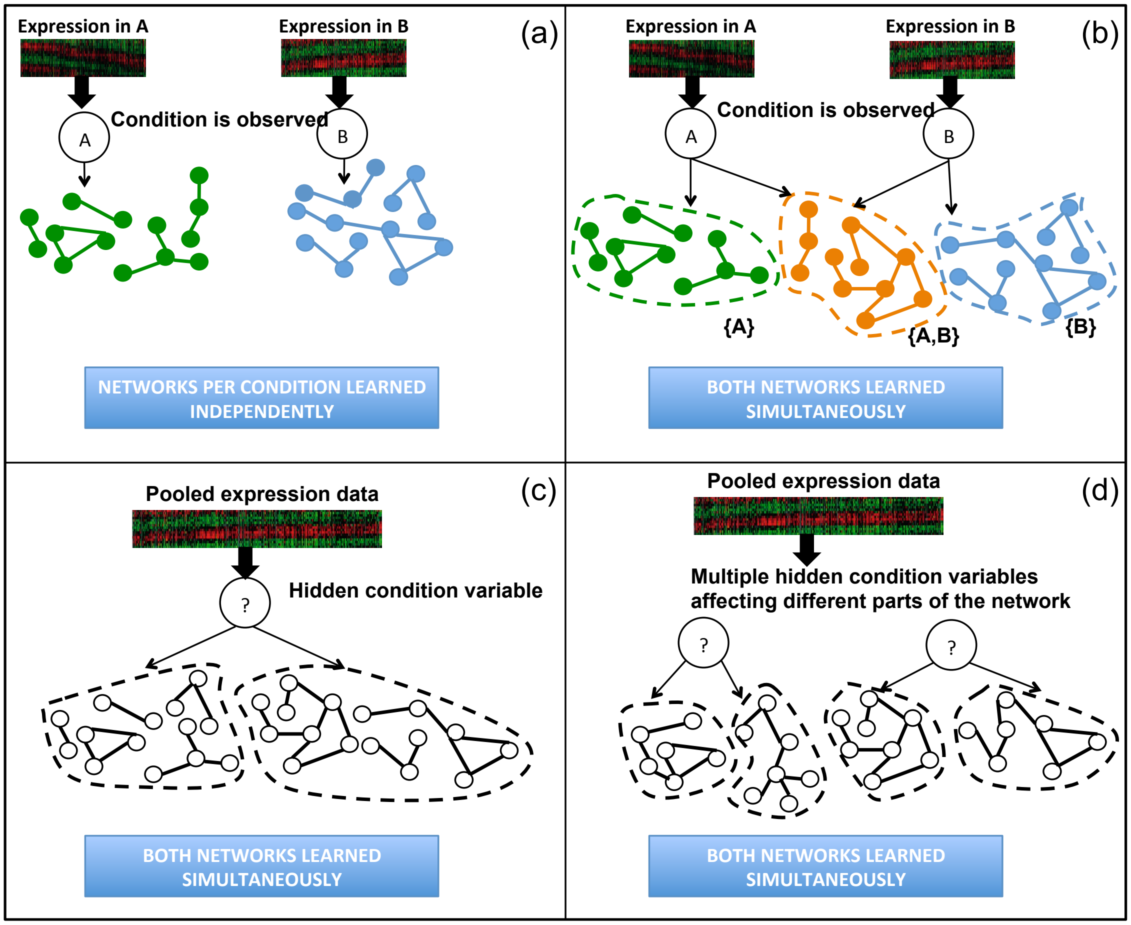

Expression in A
Expression in B
(a)
Condition is observed
A
B
NETWORKS PER CONDITION LEARNED INDEPENDENTLY
Expression in A
Expression in B
(b)
Condition is observed
A
B
{A}
{A,B}
{B}
BOTH NETWORKS LEARNED SIMULTANEOUSLY
Pooled expression data
(c)
Hidden condition variable
?
BOTH NETWORKS LEARNED SIMULTANEOUSLY
Pooled expression data
(d)
Multiple hidden condition variables affecting different parts of the network
?
?
BOTH NETWORKS LEARNED SIMULTANEOUSLY

Figure 1.3: Different formulations of condition-specific network learning. Different panels represent types of condition-specific network learning. Panels (a) and (b) show models with observed condition variable. Colors in panels (a) and (b) discriminate the networks for each condition. Orange is for the shared part. Panels (c) and (d) show models where the condition variable is hidden. Panel (a) is the simplest case of observed variable, and networks per condition are learned independently. Panel (b) represents an extension to multi-nets where we pool data to better incorporate shared information across conditions. Panel (c) is the simplest case of hidden condition variable, with a single condition variable influencing the entire network. The black-white network shows the union of the two condition-specific networks, and different assignments of the condition variable selects different parts of the union graph demarcated by the dashed lines. Panel (d) represents a more complex type of hidden condition variable setting. Here different connected components of the union graph are controlled by different condition variables.

Condition-specific networks of different granularities We consider the case where the condition variable is inferred (Fig 1.3, Panels (c) and (d)). In this situation, it is unclear if having a single condition variable for the complete data point (joint assignment to all random variables) captures the finer subtleties of condition-specific response. It is likely that for a given data point some values may be generated from the shared part of the network, whereas some values may be generated from specific parts of the network. This kind of fine-grained sharing requires us to have multiple hidden variables per datapoint. A related issue to consider is if, and, how the underlying graph topology constrains the number and values of condition variables. Do more connected graphs favor some types of mixture models, and sparsely connected graphs favor other models?

Statistical validation The problem of inferring networks from expression data whether in, one or multiple conditions, is essentially an instance of unsupervised learning. This raises the important question of how we can validate the models we learn from data. This question is especially difficult for network learning because the data may be equally likely to be generated from multiple networks. How then can we evaluate the structures we learn? Because probabilistic models have both a structural and functional component, how can we account for the functional component during method comparison? It is common practice to use data likelihood as a yard-stick for comparison, but what can we use if we do not have the ability to compute likelihood? Do approximations of likelihood, such as pseudo likelihood serve as effective measures of comparison?

Identifying the target problem A computational method is of little use if it is not demonstrated on a real-world dataset. In our setting, we need to demonstrate the practical utility of our condition-specific network learning problem on a real biological problem. Because we want to learn multiple networks, one per condition, irrespective of data sharing or not, we need enough samples per condition so as to reliably learn these networks. We therefore need to identify suitable datasets that satisfy our problem description, and that can lead to biological discovery on applying our algorithms.

1.3 Computational contributions of this dissertation

This dissertation solves the above challenges by accomplishing the following goals:

- Develops an algorithm for learning the structure of general undirected probabilistic graphical models and extensions to it for efficiently learning large networks.
- Extends existing multi-net learning frameworks to learn multiple networks in a more data-efficient manner, and in an undirected graphical modeling framework.
- Develops an algorithm for learning a mixture model of general undirected graphs.
- Develops algorithms for inferring condition assignments at different levels of granularity.
- Empirically analyzes the performance of mixture models of condition-specific networks as function of the underlying graph topology.
- Empirically analyzes algorithms learning pairwise versus higher-order dependencies.
- Develops a validation framework for evaluating the quality of inferred networks using both structure of the network and the functions computed by the network.

1.4 Biological challenges of condition-specific networks

Learning condition-specific networks for describing system wide condition-specific response poses the following biological challenges and questions:

Constraints in networks representing condition-specific response Modularity is a design principle of most cellular networks that allows a cell to achieve its myriad functions by re-using different combinations of these components [101, 110, 75]. This component re-usability in cells imposes constraints on the structure of the networks that must be incorporated within our network learning framework. Imposing such constraints are especially important for learning networks from biological datasets where data sparsity is a norm more than an exception. In particular, small perturbations to the data can result in dramatically different network structures, producing many more differences than what are likely to be biologically meaningful, and almost always we will incorrectly conclude that networks from different conditions have undergone a non-trivial amount of re-wiring. Hence, to accurately identify the network structure, including the networks parts that are unique and shared across the different conditions, it is crucial for our algorithms to incorporate and exploit these constraints.

Validation of results The problem of validation also arises in a biological context. Unlike statistical validation which assesses the robustness and non-randomness of our methods, this type of validation assesses quality of the inferred networks based on the number of biologically meaningful dependencies present in the network. This relies on biological knowledge of genes and includes validating both pairwise dependencies as well higher-order dependencies. Because we consider multiple biological categories, this naturally raises issues of multiple hypothesis testing [130]. How do we assess the statistical significance of our results in the face of multiple hypotheses? Further, the significance of our results depends upon our definition of a background or null distribution [20]. How do we generate realistic models of the null distribution such that we have high recall and without sacrificing on precision and false discovery rates?

Interpretation of results Our ultimate goal is to get a better understanding of how cells behave under different conditions. Given that there is prior domain knowledge about the behavior of the cells under certain conditions, a central challenge is to integrate the new information conveyed in network structures with the existing knowledge. This is not straightforward because our networks use only gene expression data, which may not necessarily represent the behavior at proteomic and metabolic levels [56]. Further, the prior knowledge is available at different levels of organization of the cell, ranging from cellular phenotype (such as cell size, density and granularity), to precise information of key genes involved in the pathways. How do we interpret our results such that it fits and extends the overall picture of cellular behavior under a particular condition?

Condition-specific networks for yeast stationary phase Yeast stationary phase is a state of cultures where there is no net change in the culture density and is induced by starving cells for glucose [55]. Contrary to "stationary", which has a danger of implying that nothing is happening in stationary phase cultures, this phase is a highly regulated, and conserved process [4, 94]. Recent research has shown that yeast stationary phase cultures have a complex heterogeneous population structure composed of at least two types of cells, quiescent and non-quiescent, that differ at the morphological and expression levels [3, 5]. How can we obtain a finer understanding of the differentiation processes of these cells, that captures information beyond differential expression? Can we learn quiescent and non-quiescent networks to identify fine-grained functional interactions among genes using expression data from these cells? Do the similarities and differences at the network level of two cell populations provide insight into the differentiation mechanisms of these cells?

Species-specific networks: another example of condition-specific networks? Identification of evolutionarily conserved signals at the DNA level has been one of the important questions of comparative genomics [76]. Evolutionary conservation at the sequence level provides only one aspect of the multitude of information in living systems, and this was demonstrated by Stuart *et al* and Bergmann *et al.* who identified conserved co-expression relationships among orthologous pairs of genes [131, 13]. However, such functional relationships need not be restricted to pairs and co-expression. Can our condition-specific network learning algorithms identify higher-order functional relationships that are conserved and specific to different species?

1.5 Biological contributions of this dissertation

This dissertation addresses the above challenges by accomplishing the following goals:

- Develops an approach for learning networks, not only for one condition but for any subset of conditions. This allows us to robustly identify both shared and unique components of condition-specic cellular networks.

- Draws upon expert evidence for interpretation of results. Our model organism for the cell population case is yeast, and for the species-specific networks are fly and yeast. Both these organisms are well-studied and have associated organism-specific databases that we leverage to interpret results to construct the big picture.

- Applies our condition-specific network learning algorithms to two yeast cell populations to understand how two morphologically dissimilar populations respond to the same starvation stress. We identified several functional interactions that suggest respiration-related processes are shared across the two conditions. We also identified interactions specific to each population including regulation of epigenetic expression in the quiescent population, consistent with known characteristics of these cells. We found several high confidence cases of combinatorial interaction among single gene deletions that can be experimentally tested using double gene knockouts, and contribute to our understanding of differentiated cell populations in yeast stationary phase.

- Applies our condition-specific network learning algorithms to microarray datasets from two species: yeast and fly. Analysis of the inferred networks from the two species suggests that sharing patterns in networks from two species are much more complex than in networks from two populations of the same species. Our results from comparing species-specific networks suggest that basic metabolic processes (fatty acid oxidation, pyruvate metabolism, IMP bio-synthesis) are conserved between yeast and fly, and the processes unique to each species include fungi-specific (filamentous growth) and higher animal-specific (oocyte fate determination) functions. These results are consistent with existing biological knowledge and also include novel discoveries.

1.6 Organization of this dissertation

This dissertation is organized as follows: Chapter 2 performs a literature survey of the existing work on condition-specific networks. This includes related work on both machine learning approaches for learning networks, which may not have been necessarily applied to biological domains, and work specifically focussing on networks in biology. We also give background material on biological networks and probabilistic graphical models. Chapter 3 focuses on validation of network structure, both statistically and biologically. We describe simulation frameworks that generate realistic data from networks of known topology and that can be used for performance comparison of different algorithms. Chapter 4 discusses the problem of learning a single network using an unsupervised structure learning algorithm. We compare our algorithm against several state-of-the-art network learning algorithms for both directed and undirected networks and demonstrate that our algorithm outperforms the majority of the compared algorithms. We discuss choices of different conditional probability distributions, which are important modeling questions to address before we tackle the general problem of condition-specific network learning. We then describe an enhancement to the basic learning framework that can efficiently learn genome-wide networks.

Chapter 5 performs an in-depth analysis of the importance of higher-order dependencies in biological networks. In particular, we identify characteristic of the graph structure that indicate higher-order dependencies. Chapter 6 describes approaches for learning condition-specific network models that provide a network view of condition-specific response. The first part of the chapter describes a simple approach to condition-specific network analysis. In particular, we show that we can learn separate networks using data from two conditions and perform comparative analysis to identify the similarities and differences in the networks. Although this approach finds meaningful dependencies, it is a first cut approach, which does not exploit shared information across conditions. We then consider a more general formulation of condition-specific network learning and describe models with both observed and unobserved conditions. We also consider several version of models with the unobserved condition, which vary in the granularity of the condition variable affecting gene expression. We empirically analyze the effect of the underlying network topology on the performance of different models.

Chapter 7 describes application of our different condition-specific network learning algorithms to address two problems in biology: learning condition-specific networks for two yeast populations, learning species-specific networks for the yeast and fly species. We conclude in Chapter 8 summarizing the main contributions of this work and propose future extensions.

Chapter 2

Background and related work

The work described in this dissertation is based on two important ideas: (a) inference and analysis of biological networks, and (b) analysis of condition-specific response. Both these ideas have been one of the core themes in systems biology and molecular cell biology and therefore a vast amount of literature is related to this dissertation. An in-depth survey of all computational and biological approaches of network biology and condition-specific specific response is beyond the scope of this disseration. Instead, we focus on the most relevant areas: the semantics of cellular networks, computational approaches for learning networks from data, and the connection between network models and condition-specific behavior.

2.1 Cellular networks

Living cells are complex, dynamic systems that respond to changing environmental conditions by integrating and processing information at various levels of intra-cellular organization. Systems biology views this information processing system in cells as a network of bio-chemical entities (genes, proteins and metabolites) that interact and exchange informa-

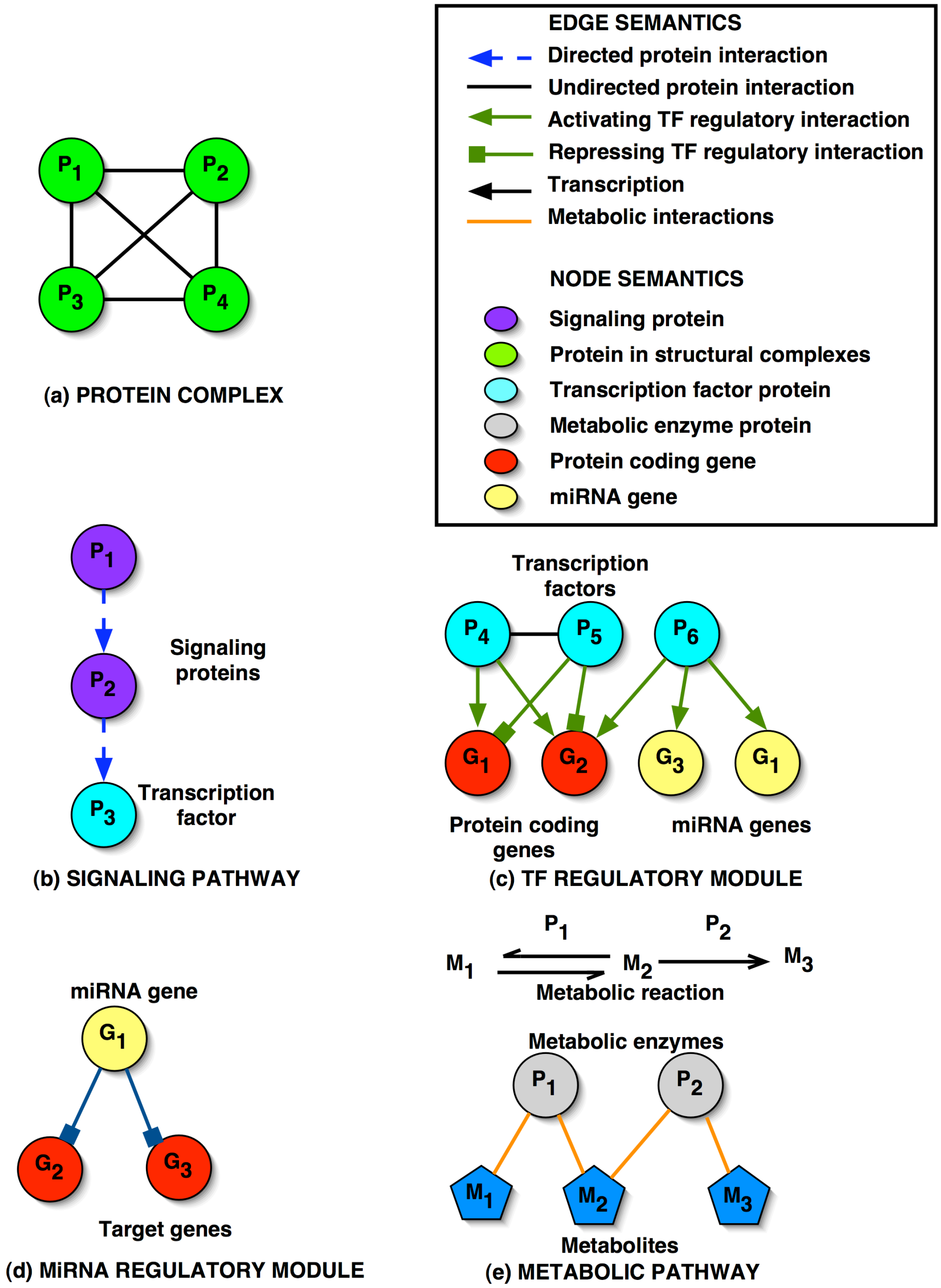


Figure 2.1: Biological modules representing the building blocks of condition-specific networks

tion, in a time and space-dependent manner to accomplish the myriad cellular functions.

Networks may be of different types based on the class of nodes that are interacting (protein interaction networks versus regulatory networks), or on the basis of the edge semantics (physical versus functional networks). We first describe the different types of networks based on the class of nodes and then describe which of these can be considered as physical versus functional interaction networks.

2.1.1 Network classification: node type

Protein interaction networks

A protein complex is a set of two or more proteins that physically interact to accomplish a specific biological function. In network terminology, a protein complex is described as a clique, with proteins as the nodes and the protein interactions as undirected edges (Fig. 2.1 (a)). For a protein complex to be clique, it must be the case that all proteins in the complex interact with each other. Since this is not always true, a protein complex is often described as a dense subgraph with a large number of interactions among the proteins.

Transcription factor complexes are well-known examples of protein complexes. In yeast there are approximately 200 transcription factor complexes [59]. Other protein complexes form structural units in the cell responsible for specific functions such as translation, RNA processing, cellular transport etc [59, 126].

Regulatory networks

A regulatory module is a building block of the regulatory network and is composed of a set of transcription factors that regulate a set of common target genes (Fig. 2.1(c)). The transcription factors regulate their target genes by binding to the $5'$ region upstream of the coding region. Some of the genes may code for transcription factor proteins and others may produce non-coding RNA that eventually is processed into microRNA.

Regulatory module (TF based): A regulatory module is composed of two types of nodes, transcription factor protein nodes and target gene nodes. The edges of the regulatory module are also of two types, undirected edges between the transcription factors indicating the formation of a transcription factor complex, and directed edges from transcription factors to target genes. The edges between the transcription factors are optional. The edges between the transcription factor and the target genes also have a sign, which indicates the type of regulation of the transcription factor on the genes. The type of regulation is activating or repressing.

Both the type of regulation and the number of transcription factors regulating the target genes in the regulatory module varies with the environmental conditions [61]. For example the Rap1p transcription factor in yeast is known to bind different targets under diferent environmental conditions [22]. As a result the regulation of the transcription factors on the target genes is highly combinatorial in nature.

An example of a regulatory unit with interactions between the transcription factors is the SAGA complex in yeast, composed of 17 proteins that must interact to start transcription. Exams of regulatory units that do not have protein interactions include transcription factors that simultaneously bind to different binding sites of the target genes such as the MCM1 and STE12 transcription factors.

Regulatory module (miRNA based): These regulatory modules also capture gene expression regulation, but, at the post-transcriptional level (Fig. 2.1(d)). A transcription factor acts on the gene's DNA sequence to allow or inhibit transcription, whereas, a miRNA acts on the gene's mRNA product, which is produced after transcription. The nodes in these modules are the regulating microRNA and the target genes, and the edges are the regulatory interactions. The regulatory interactions are typically repressive in nature, although, we cannot rule out activating actions.

Metabolic networks

A metabolic pathway is composed of a set of metabolic reactions that constitute a small part of larger metabolic pathways such as the yeast galactose pathway or the nitrogen metabolism pathway. The role of a metabolic enzyme is to catalyze a particular metabolic reaction that converts a substrate metabolite to a product metabolite. There are two ways of describing metabolic networks. In the first, nodes in the metabolic pathway are the metabolic enzymes and the metabolites (Fig. 2.1(e)). The edges are inserted between a metabolic enzyme and a metabolite if the metabolite is either a substrate or a product of a reaction catalyzed by the metabolic enzyme. In the second, nodes are only metabolic enzymes and edges are inserted between two nodes if the product of one enzyme is the substrate of the other.

Signaling networks

Signaling pathways, also known as signal transduction pathways, are linear or tree-like connections of proteins, which are responsible for transmitting changes in the extracellular environment to the regulatory network (Fig. 2.1(b)) [128]. Similar to a protein complex, a signaling pathway is composed of proteins as nodes and interactions as edges. However, the edges of a signaling pathway are directed, indicating the flow of information.

Signaling pathways are composed of a membrane protein on one end and a transcription factor on the other end. The membrane proteins monitor the extracellular environment and send signals to proteins downstream via post-translational modifications, terminating in the activation of transcription factor. These transcription factor proteins then bind to upstream regions of their target genes and cause their activation or repression. Examples of such pathways include the yeast pheromone response pathway, cell wall integrity pathway (PKC), hyperosmolar pathways (HOG), mitogen activated protein kinase pathways (MAPK) [128, 97].

2.1.2 Network classification: edge semantics

Networks can also be classified based on the semantics of the edge. There are two main types of networks based on edge semantics: (a) physical interactions, (b) functional interactions. Physical interaction networks are those where the edge represents a physical contact among the interaction nodes, for example, in protein interaction (especially pairwise interactions) networks, and transcriptional regulatory interactions which represent interactions from a transcription factor protein and a target gene. Functional interactions describe a functional rather than a physical mechanism of interaction. Such functional interactions may be a cascade of physical interactions, or parallel pathways that must function in concert. In general, functional networks represent a wide range of interactions including regulatory, protein and metabolic interactions. Two well-known examples of functional networks are genetic networks and expression networks.

Genetic networks are networks where an edge is indicative of a significant, often lethal, phenotype that occurs only when both genes are knocked out, but not when either one is knocked out. Expression networks are by far the most ubiquitous type of networks that are constructed using microarray-based expression profiles of genes. These networks can be grouped into co-expression networks, where edges represent statistical correlation (in terms of Pearson's or Spearman's correlation coefficient), and dependency networks, where edges represent general statistical dependencies such as mutual information which represents both correlation and anti-correlation. Dependency networks can be further classified into those that have pairwise dependencies (among two genes), and those that have higher-order (among more than than two genes) and pairwise dependencies.

2.2 Condition-specific response

Condition-specific response is a genome-wide phenomenon which involves almost all layers of organization in the cell. To fully characterize how a cell behaves under a condition we need to measure the genes, proteins and the metabolites within a cell. However, for the vast majority of conditions, we have only gene expression data because proteome-wide and metabolome-wide measurements are much harder to obtain. Our methods focus on describing condition-specific behavior using gene expression data. We realize that this is incomplete, but our methods are easily applicable to protein and metabolite expression as they become available.

2.2.1 Network-based methods of condition-specific response

The simplest approach of characterizing condition-specific response is to identify the set of differentially expressed genes in the condition of interest. However differential expression analysis only identifies a list of genes, and were originally developed for situations with a few measurements per gene. Advancement in high-throughput sequencing technologies such as RNASeq [139], in collaboration with the ModENCODE project (`http://www.modencode.org/`) are producing massive amounts of data per tissue and per condition. While differentially expressed genes are still informative, these datasets give us the ability to extract more complex patterns of interactions, beyond differentially expressed lists of genes. The advantages of network-based description of condition-specific responses was recently demonstrated by Chuang *et al.* who identified disease-specific subgraphs that included genes that were not necessarily differentially expressed [30]. We now describe approaches for capturing condition-specific response that incorporate network structure of genes at some stage of their analysis.

One example of a comprehensive analysis of transcription factor activity under different environmental conditions has been done by Harbison *et al.* [61]. This analysis used ChIP-chip analysis to study the binding activities of 85 transcription factors in yeast under twelve different environmental conditions. By incorporating different environmental conditions, the authors discovered several condition-specific patterns for transcription factor regulation. Although, this work provided deep insight into the effect of environmental conditions to binding patterns of transcription factors, it did not take into account the transcript abundance, which is important to understand the effect of different binding patterns on the dynamics of gene expression under different environmental conditions. Furthermore, this focuses on the re-wiring of the transcriptional regulatory network, whereas it is important to consider re-wiring at the metabolic or more general functional network level.

McCord *et al.* combined gene expression with transcription factor binding data to study condition-specific behaviour of transcription regulation under a large number of environmental conditions [98]. Using a novel protein binding array technology, they identified a large number of target genes for yeast transcription factors. The differential expression of the target genes under different environmental conditions was then used to predict the conditions in which a transcription factor is likely to be active. Although, this approach incorporates gene expression to predict condition-specific functions of transcription factors, it also is targeted to the transcriptional regulatory network.

Purely computational approaches are based on bi-clustering, which cluster genes and conditions simultaneously [133, 14, 36, 18], and identify sets of genes that are co-regulated in sets of conditions. More advanced approaches additionally identify transcription modules (set of transcription factors regulating a set of target genes) that are co-expressed in a condition-specific manner [9, 123, 10, 51]. The GRAM algorithm by BarJoseph *et al.* infers transcriptional modules, which have simple network topologies (bipartite graphs) with edges from transcription factors to target genes. Segal *et al.*'s COPR model uses module networks to build transcriptional programs with slightly more complex topology (decision trees). However, these approaches focus on the transcriptional regulatory network, and do not provide fine-grained interaction structure that explains the condition-specific response of genes.

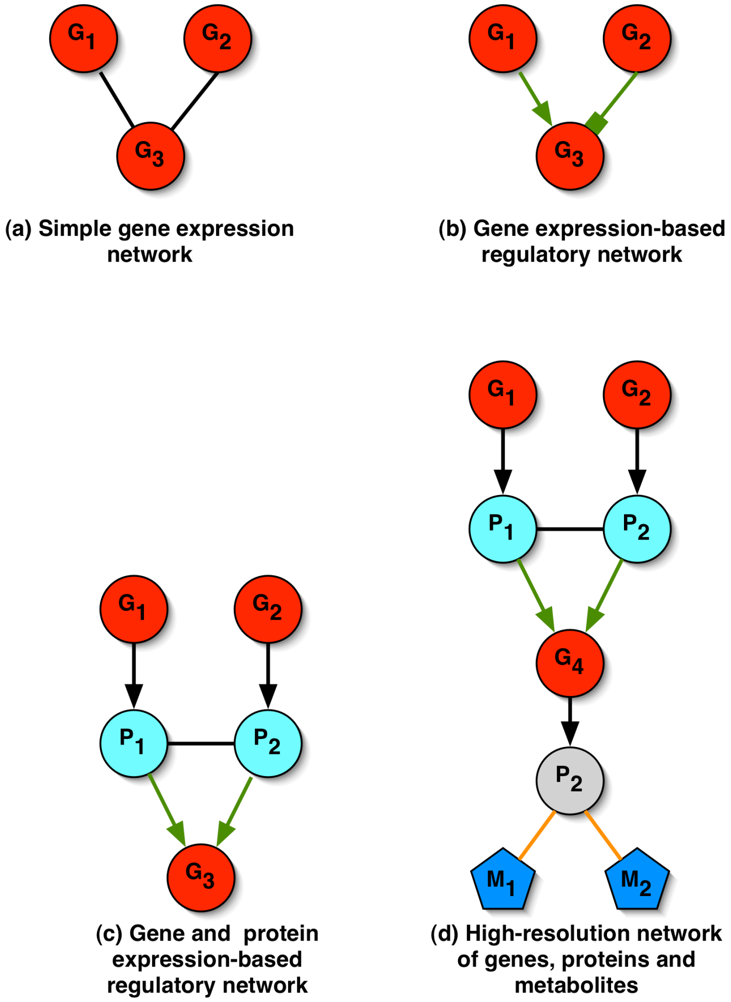
Rachlin *et al.* described a new graphical construct to add context information in the analysis of protein interaction networks [111]. These "context-specific" networks make use of biological annotation of the protein functions to provide contextual meaning to the protein interactions. However, this approach does not account for gene and protein expression and is focussed to the protein interaction layer.

Other approaches by Kim *et al.* [77] and Tuck *et al.* [136] explicitly capture the condition-specific activity of the edges in the transcriptional regulatory networks. The former focuses on microarray data of yeast cells under different environmental conditions, whereas the latter focuses on microarray data from different healthy and cancer human cells. However, these models used a discriminative rather than a generative approach to highlight the condition-specific behaviour of the networks and require that the condition variable be known for each dataset. Discriminative approaches also highlight the differences between, whereas identification of similarities is equally important.

Approaches for differential expression analysis have also started incorporating the network structure. The graph-based iterative group analysis method constructs a bi-partite *evidence graph* [21], where nodes represent both genes and Gene Ontology annotation categories. Edges connect genes to annotation category nodes. This is collapsed into a simple graph comprising only gene nodes by adding edges between two genes connected to the same category node. Each node is assigned a rank inversely related to differential expression, followed by identification of local minima nodes that all have lower ranks than their immediate neighborhoods. The algorithm then proceeds iteratively from the local minima nodes to include nodes with the smallest rank in the neighborhood, and computing a hypergeometric p-value assessing the statistical significance of the minimum rank of the subgraphs. The subgraph with the smallest p-value is considered as differentially regulated and the procedure is repeated for all local minima nodes. This approach is attractive because of the direct incorporation of annotation information during differential enrichment analysis. However, this relies on the existing literature annotation and no network learning is performed.

A related approach is Guido *et al*'s the Mixture Model on Graphs [119], which uses a Bayesian framework to infer the posterior probability of the differentially enriched subgraphs. The main idea here is to consider a mixture model for three components, one for highly expressed, one for lowly expressed and one for no change. Gene expression of each gene is assumed to be generated from this mixture model. The genes themselves are connected in a network of known structure, which is incorporated into a prior probability distribution of the hidden class variable. Specifically, the prior probability is a multinomial vector estimated from a weighted, regularized sum of the class assignments of the neighbors. The likelihood model uses the mixture model to generate the observed expression. This approach also assumes the graph is given and focuses on differential expression in experimental versus test conditions.

Recently the differential dependency network approach was proposed by Zhang *et al.* [146] which identifies differentially expressed neighborhoods of genes. This approach identifies multiple sufficiently good neighborhoods per gene for each condition separately. Differential expression of a neighborhood is measured by the difference in the coefficient of determination estimated on the two datasets using the same neighborhood. The coefficient of determination approximates the reduction in the variance of gene expression by the neighborhood. A neighborhood that is highly differentially expressed will have a significantly different coefficient of determination on the two datasets. A permutation test-based strategy is used to assess statistical significance of differential expression. Although this approach finds networks, it is focused on differential expression. Further the permutation strategy and the algorithm itself are very computationally intensive requiring enumeration of all subsets of the complete random variable set up to size K.



(a) Simple gene expression network

(b) Gene expression-based regulatory network

(c) Gene and protein expression-based regulatory network

(d) High-resolution network of genes, proteins and metabolites

Figure 2.2: (a) A pure gene expression network. Here edges mean a functional relationshop. (b) A gene regulatory network, where the edges imply a regulatory action of genes coding for transcription factor proteins. (c) A gene, protein expression network. Here edges imply a regulatory action (Protein-DNA) or a undirected protein interaction. (d) A gene, protein, metabolite network. This is the highest resolution network, integrating the different layers of condition-specific networks.

2.3 Approaches of network learning

The structural and functional aspects of biological networks are usually unknown for different environmental conditions. Algorithms for biological network inference aim to reconstruct the structural and functional components of networks using measured attributes of the network nodes. The *resolution* of an inferred biological network is defined as the extent to which the network captures the structural and functional aspects of the true network. We group algorithms used for network inference based on the resolution of the networks inferred by the algorithms.

A high-resolution network is one in which the network edges have a direct physical interpretation, such as a protein-protein interaction or a protein-DNA interaction. The edges of a high-resolution network have several physical attributes such as edge directionality, edge sign etc. A low-resolution network is one in which the network edges represent functional or phenomenological relationships between genes, without explicitly accounting for the physical interpretation of these relationships.

The simplest type of inferred network is a gene expression network, which is constructed de-novo from gene expression data (Fig. 2.2(a)). The nodes correspond to the genes and the edges represent a functional interaction between two genes, where the strength of interaction is quantified by a measure of statistical dependency. These networks are considered low resolution because the edges indicate functional interactions, which is an approximation of some complex bio-chemical phenomenon such as protein interactions or cascades of signaling reactions. These networks provide a high-level, system-wide view of the interaction pattern among genes.

Bayesian networks[1] are directed, probabilistic, graphical models and are well known models for generating networks at this level of resolution [43, 127, 143, 109]. Although the edges of a Bayesian network are directed, the directionality usually implies correlation and not causality. Other examples of this class of models are Boolean networks [93], differential equation models [18], Algorithm for Accurate Reconstruction of Cellular Networks (ARACNE) [91], regression models [137, 46], conditional-correlation models [72]

[1] Bayesian networks are a very general class of probabilistic graphical models, and de novo network inference is one of the many ways in which Bayesian networks may be used. It is possible to infer higher resolution models using Bayesian networks also.

and graphical Gaussian models [140]. These models differ in the type of dependency they capture, the amount of prior information about network structure that can be incorporated, and the values that can be taken by the network nodes. ARACNE, for instance uses mutual information between pairs of random variables denoting genes, whereas all the other models can capture higher order dependencies. Boolean networks require that nodes take only binary values, whereas regression models require that the nodes take continuous values. Finally, Bayesian networks can incorporate a large amount of prior information about the network structures. In fact, Bayesian networks are a very general class of models and incorporate most of the features of the other network reconstruction algorithms in this category. However, they do not perform quite as well in tasks that other algorithms are designed to optimize. For example, if we are interested in only pairwise dependencies then ARACNE is much better and faster than Bayesian networks.

The next class of algorithms incorporate other sources of data to provide more semantics to the edges than de novo construction (Fig 2.2(b)). The nodes still correspond to genes, but the edges imply a regulatory action, which is either activating or repressing. Gene regulatory networks are examples of networks inferred at this level of resolution. The models that infer networks with this level of resolution use protein-protein and protein-DNA interaction data, in addition to gene expression data. Examples of these models are Yeang *et al.*'s physical network model for explaining perturbations in the gene and metabolic layers [142]. These models are based on undirected graphical models, factor graphs, that use physical interaction data to explain perturbations at the regulatory and metabolic levels. Although, this model infers a large number of physical attributes such as edge directionality and sign, it does not incorporate protein expression, and assumes that the structure of the network is essentially fixed.

Other examples of detailed models include Genomica and its successor, the co-regulated overlapping processes (COPR) model [123, 10]. These models are based on module networks, which is a class of directed graphical models to capture relational data [125]. In these models, edges are inferred between transcription factors and target genes. These models combine known transcription factors with transcription factor binding site sequences to build detailed regulatory networks. However, these models do not account for protein expression and are targeted toward the regulatory network rather than the complete physical network.

The next level of models incorporate gene and protein expression in addition to known protein-protein and protein-DNA interactions (Fig. 2.2(c)). Learning these models is difficult because protein expression is not available for most experimental conditions. However, there are some approaches of inferring the expression of transcription factor proteins from the gene expression of target genes [81, 50].

Although, there are very few models that combine protein and gene expression to infer networks, there are signaling networks that have been inferred using solely protein expression data [141, 116]. However, these models are targeted to specific signaling pathways triggered in human cells, for which it is possible to measure phosphorylation levels of signaling proteins.

The most detailed class of networks incorporate expression information at the protein, gene and metabolite levels and physical interaction information such as protein-protein, protein-DNA microRNA-mRNA interactions (Fig. 2.2(d)). Gat-Viks *et al.* have proposed a factor graph based model that makes extensive use of biological literature to build a detailed network for a particular biological pathway: yeast osmotic stress pathway [49]. This approach refines regulatory functions capturing relationships between transcription factors and target genes and uses inference techniques to estimate protein expression. Although, this method develops highly detailed networks, it requires the basic network structure to be determined from literature. Because of its heavy reliance on literature, it cannot be used for all biological pathways, and therefore cannot be used to infer the complete network.

2.4 Modeling biological networks as probabilistic graphical models

Mathematical models for representing condition-specific networks, must have the following properties: (a) capture different types of dependencies among network nodes including pairwise, cyclic and higher-order dependencies, (b) account for noise and uncertainties in the data, (c) provide a generative framework characterizing the space of node values under a specific environmental condition, (d) allow tractable computation of global functions (such as joint probability distributions) over the complete condition-specific network. Using such a generative framework we can generate and predict expression states of the network nodes under different perturbations.

Biological networks have been represented using different mathematical models, such as boolean networks [92], regression models [46] and probabilistic graphical models (PGMs) [40, 123, 91]. See [16, 34] for reviews. We focus on PGMs because similar to boolean networks and regression models, they capture the structual and functional aspects of networks. However, PGMs offer additional flexibility of using either discrete or continuous variables and provide well-founded semantics for handling uncertainty and noise.

A probabilistic graphical model is associated with two components: a graph $\mathbf{G}$ and a set of functions $\mathcal{F}$. Nodes represent genes or proteins and random variables encode the expression level of the nodes. The edges of $\mathbf{G}$ describes the statistical dependency structure among the network nodes. A function $f_i \in \mathcal{F}$ describes the functional relationship between a random variable X_i and its neighbors. The two well known classes of PGMs are directed graphical models and undirected graphical models.

Learning in PGMs includes two tasks: parameter estimation on a fixed network structure, and structure learning where both the structure and the parameters of the model need to be estimated. Learning the parameters given the structure of the network is tractable for directed graphical models. Learning the structure for PGMs is NP hard as it involves a search over the space of possible graphs. This space is super-exponential $(O(2^{\binom{N}{2}}))$ in N, the number of nodes in $\mathbf{G}$ [28]. However, greedy search algorithms have been proposed that find a structure optimizing a likelihood-based score describing the model fit to the data [44]. These scores usually correspond to the posterior likelihood of the model given the observed data:

$$P(\mathbf{G}|\mathcal{D}) \propto P(\mathcal{D}|\mathbf{G})P(\mathbf{G}) = P(\mathbf{G}) \int P(\mathcal{D}|\theta, \mathbf{G})P(\theta|\mathbf{G})d\theta$$

Here, $\mathcal{D}$ is the observed data, $P(\mathbf{G})$ specifies a prior probability distribution on the graph structure, $P(\mathcal{D}|\theta, \mathbf{G})$ is the likelihood of the data given the model, θ is the set of parameters associated with $\mathcal{F}$. Depending upon the parametric form of the prior and θ, different scores can be derived. These scores often introduce a regularization term for penalizing complex structures.

In directed graphical models such as Bayesian networks, the functions f_i describe the conditional distribution of X_i given its parent variables. The global joint distribution of all random variables is given by the product of each f_i. This decomposability of joint distribution function over the local f_i makes Bayesian networks a very convenient model for biological networks because the complete structure can be learned by optimizing the local f_is. The f_is in turn can capture pairwise and higher-order dependencies. For completely observed data, Bayesian score-based structure learning algorithms are tractable and yield reasonably good solutions. Examples of such scores include the Bayesian Information Criterion (BIC) [19, 132] and Bayesian Dirichlet (BDe) scores [31, 65]. Unfortunately, Bayesian networks cannot explicitly cyclic dependencies, which is limiting for modeling biological networks.

In undirected graphical models, the f_i correspond to potential functions, one for each clique in $\mathbf{G}$. Unlike directed models, the product of these potentials does not yield a consistent joint distribution. A valid joint distribution requires the computation of a normalization factor called the partition function. Computation of the partition function for the general class of undirected models is computationally intractable (NP-hard) [1]. As a result both parameter estimation and structure learning in these models is hard. The learning problem is made tractable by imposing constraints on the structure or by using scores other than likelihood such as pseudo likelihood [15, 68, 1]. Examples of these models with tractable learning algorithms include limiting the maximal degree of a node in a factor graph [1], bounding the tree-width of a Markov network [104], constraining the graph structure to be a tree [29], or estimation of lower-order (often pairwise) functions [91]. Pairwise models are also attractive because they scale to networks of several thousands of nodes. However, approximation of higher-order dependencies via pairwise functions may cause these algorithms to identify spurious dependencies.

An intermediate between directed and undirected graphical models are dependency networks [64, 121]. Similar to directed models, f_i represents a conditional distribution, and similar to undirected models, the graph structure need not be acyclic. The structure is learned by estimating the best neighborhood of each variable independently, which makes the learning problem computationally efficient. Unfortunately, the final structure may be inconsistent (i.e. X_i may be in X_j's neighborhood but X_j may not be in X_i's neighborhood) and may not produce a consistent joint probability distribution. Although, in the limit of infinite data it is possible to learn both consistent structures and distributions, the biological domain almost always suffers from limited data.

Chapter 3

Evaluation framework of unsupervised learning of biological networks

The problem of learning networks from expression data is essentially unsupervised in nature. This means that there is no known ground truth against which we can compare to assess if our learning algorithm was able to learn the network correctly. To address this problem it is common to use a simulation framework where data (expression of genes) is generated from a network of known topology. Learning algorithms are applied to the simulated data and the inferred networks are compared to the original network to assess how well the algorithms modeled the relationships in the data. This raises two important issues: (a) generation of a realistic simulation framework for generating data that is close to the domain of interest: biological networks, (b) scores for measuring the similarity between the true and inferred networks.

This chapter addresses these two questions. We first describe regulatory network simulator which generates simulated networks as well as the data from these networks. We then describe various measures for comparing how close the inferred and true networks are. We then describe methods of assessing the quality of inferred networks in biology using existing biological annotation.

3.1 Simulation framework for generating realistic data

With the increasing availability of genome-scale data, a plethora of algorithms are being developed to infer regulatory networks [8]. Because of the absence of “ground truth” of regulatory network topology, these algorithms are evaluated on artificial networks generated via network simulators [120, 100, 91, 79].

Since gene regulation is a dynamic process, existing network simulations employ systems of ordinary differential equations (ODEs) that describe the kinetics of mRNA and protein concentrations as a function of time. Some approaches construct highly detailed models, but require large amounts of user-specified information [120, 79]. Other approaches generate large networks but use simpler models by making the mRNA concentration of target genes dependent upon mRNA concentration, rather than on protein concentration of transcription factors [100]. In real biological systems, protein expression may not correlate with gene expression, especially at steady-state, due to different translation and degradation rates [12]. These approaches also do not model protein interaction edges and, the combinatorics resulting from these interactions.

We developed a regulatory network generator, REgulatory Network generator with COmbinatorial control (RENCO), that models genes and proteins as separate entities, incorporates protein-protein interations among the transcription factor proteins, and generates ODEs that explicitly capture the combinatorial control of transcription factors. RENCO accepts either pre-specified network topologies or gene counts, in which case it generates a network topology. The network topology is used to generate ODEs that capture combinatorial control among transcription factor proteins. The output from RENCO is in SBML format, compatible with existing simulators such as Copasi [67] and RANGE [88]. Time-series and steady-state expression data produced from the ODEs from our generator can be leveraged for comparative analysis of different network inference algorithms.

3.1.1 Transcriptional regulatory network generator

RENCO works in two steps: (a) generate/read the network topology, and, (b) generate the ODEs specifying the transcription kinetics. For (a) proteins are connected to each other via a scale-free network [2], and to genes via a network with exponential degree distribution [95].

Modeling combinatorial control of gene regulation

We model combinatorial control by first identifying the set of cliques, $\mathcal{C}$, up to a maximum of size t in the protein interaction network. Each clique represents a protein complex that must function together to produce the desired target regulation. A target gene, g_i is regulated by k randomly selected such cliques, where k is the indegree of the gene. These k cliques regulate g_i by binding in different combinations, thus exercising combinatorial gene regulation. We refer to the set of cliques in a combination as a *transcription factor complex* (TFC). At any time there can be several such TFCs regulating g_i. The mRNA concentration of a target gene is, therefore, a function of three types of regulation: *within-clique*, *within-complex*, and *across-complex* regulation. Within-clique regulation captures the contribution of one clique on a target gene. The within-complex regulation captures the combined contribution of all cliques in one TFC. Finally, the across-complex regulation specifies the combined contribution of different TFCs.

We now introduce the notation for ODEs generated by RENCO. $M_i(t)$ and $P_i(t)$ denote the mRNA and protein concentrations, respectively, of gene g_i, at time t. V_i^M and v_i^M denote the rate constants of mRNA synthesis and degradation of g_i. V_i^P and v_i^P denote the rate constants of protein synthesis and degradation. $\mathbf{C}_{ij}$ and $\mathbf{T}_{ij}$ denote a protein clique and a TFC respectively, associated with g_i. $\mathbf{Q}_i$ denotes the set of TFCs associated with g_i. X_{ij}, Y_{ij} and S_i specify the within-clique, within-complex and across-complex regulation on g_i.

Based on existing work [100, 120], the rate of change of mRNA concentration is the difference of synthesis and degradation of M_i: $\frac{dM_i(t)}{dt} = V_i^M S_i - v_i^M M_i(t)$. Similarly for protein concentration, $\frac{dP_i(t)}{dt} = V_i^P M_i(t) - v_i^P P_i(t)$.

The across-complex regulation, S_i is a weighted sum of contributions from $|\mathbf{Q}_i|$ TFCs: $S_i = \sum_{q=1}^{|\mathbf{Q}_i|} w_q Y_{iq}$, where w_q denotes the TFC weight. The sum models "or" behaviour of the different TFCs because all TFCs need not be active simultaneously. The within-complex regulation, Y_{ij} is a product of within-clique actions in the TFC $\mathbf{T}_{ij}$, $Y_{ij} = \prod_{c=1}^{|\mathbf{T}_{ij}|} X_{ic}$. The product models "and" behaviour of a single TFC because all proteins within a TFC must be active at the same time. Finally, the cliques per gene $\mathbf{C}_{ij}$ are randomly assigned activating or repressing roles on g_i. If $\mathbf{C}_{ij}$ is activating:

$$X_{ij} = \frac{\prod_{p=1}^{|\mathbf{C}_{ij}|} P_p(t)}{\prod_{p=1}^{|\mathbf{C}_{ij}|} Ka_{ip} + \prod_{p=1}^{|\mathbf{C}_{ij}|} P_p(t)},$$

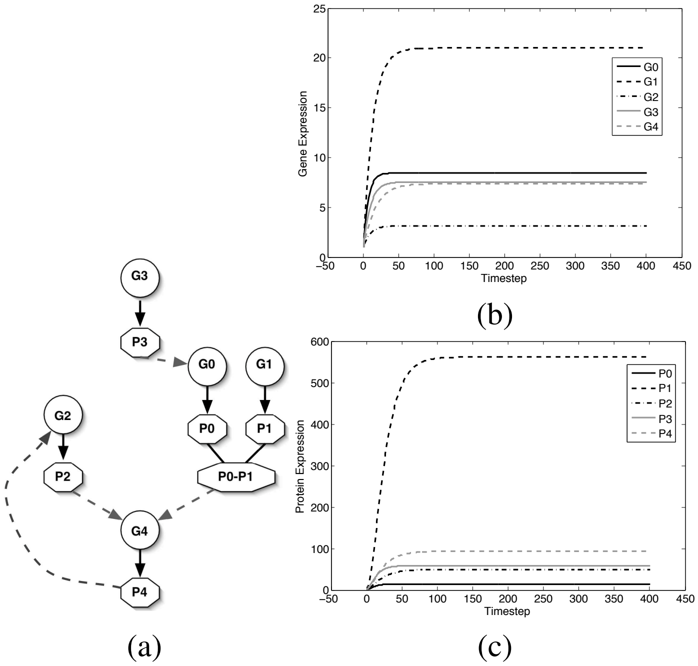
otherwise,

$$X_{ij} = \frac{\prod_{p=1}^{|\mathbf{C}_{ij}|} Ki_{ip}}{\prod_{p=1}^{|\mathbf{C}_{ij}|} Ki_{ip} + \prod_{p=1}^{|\mathbf{C}_{ij}|} P_p(t)}.$$

Ka_{ip} and Ki_{ip} are equilibrium dissociation constants of the p^{th} activator or repressor of g_i. All degradation, synthesis and dissociation constants are initialized uniformly at random from $[0.01, V_{max}]$, where V_{max} is user-specified.

3.1.2 Example network

We used RENCO to analyze : (a) mRNA and protein steady-state measurements, and, (b) combinatorial gene regulation, in a small example network.



(a) (b) (c)

Figure 3.1: (a). Example network. Dashed edges indicate regulatory actions. Wild-type gene (a) and protein (b) time courses.

Importance of modeling protein expression

The example network has 5 genes and 5 proteins (Fig 3.1 a). The gene G_4 is regulated via different combinations of the cliques $\{P_2\}, \{P_0, P_1\}$. We find that the wild-type time courses of individual mRNA expressions are correlated with corresponding proteins (Fig 3.1b and c). But because different genes and proteins have different degradation and synthesis rate constants, the mRNA population as a whole does not correlate well with the protein population (Spearman's correlation $= 0.3$). Because of the dissimilarity in the steady-state mRNA and protein expression populations, genes appearing to be differentially expressed at the mRNA level may not be differentially expressed at the protein level. This highlights the importance of modeling mRNA and protein expression as separate entities in the network.

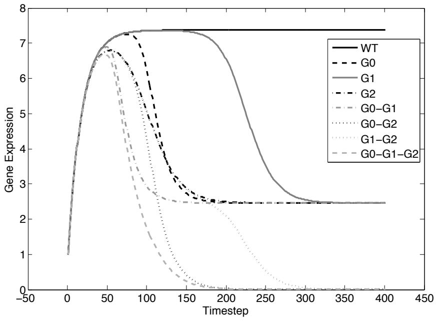


Figure 3.2: G4 time course under knock out combinations of G0, G1 and G2)

Combinatorics of gene regulation

We analyzed combinatorial control in our network by generating the G_4 time course under different knockout combinations of the G_4 activators, P_0, P_1 and P_2 (Fig 3.2). Because all the regulators are activating, G_4 is downregulated here compared to wild-type. We note that each knock out combination yields different time courses. In particular, knocking out either G_0 or G_1 in combination with G_2 is sufficient to drive the G_4 expression to 0. This phenomenon is because of the clique, P_0, P_1. This illustrates a possible combinatorial regulation process to produce a range of expression dynamics using a few transcription factors.

3.2 Structural comparison of inferred network structures

Performance comparison of different structure learning algorithms is important to identify the best algorithm. Unfortunately, this is difficult because the true network is rarely known, and therefore there is no ground truth to compare predicted structures against. Existing comparison techiques use network simulation systems to generate artificial network topologies and associated node measurements. Different inference algorithms are then used to infer networks from the node measurements and compared against the original network topology. However, scores used to compare different algorithms do not highlight the strengths and weaknesses of different algorithms because: (i) they are based on edgewise precision and recall [91] and do not account for higher-order structures involving more than two nodes, which are common in biological networks, (ii) they only compare inferred topologies without considering the functional relationships among the network nodes.

We describe two novel types of higher order scores, *pathwise* and *subgraph* score, that account for higher-order structures in the network topology (Section 3.2.1). We also describe an approach to perform functional comparison between the outputs of the structure inference algorithms (Section 3.3).

3.2.1 Topological comparison of higher-order structures

Pathwise scores

The standard precision and recall scores, referred to as "edgewise scores", are based on matching individual edges between the true and inferred network. The pathwise scores extend the edgewise scores by comparing paths and edges. The pathwise precision measures how well edges in the inferred network are supported by paths in the true network. We define pathwise precision, $P_p = \frac{1}{N}\sum_{i=1}^{N} S_i$, where N is the number of edges in the inferred network and S_i measures the support of the i^{th} inferred edge in the true network. Specifically, $S_i = \frac{1}{1+(l_i-1)}$, where l_i is the length of the shortest path in the true network between the nodes of the i^{th} edge. If no path exists, $l_i = M+1$, where M is the number of network nodes. This definition of precision allows more flexibility in cases where edges in the inferred network are not matched exactly by edges in the true network. Recall is defined in a similar way where edges in the true network are matched with paths in the inferred network.

Subgraph scores

We introduce subgraph scores to evaluate how well algorithms for network structure inference capture higher-order dependencies. Similar to the network motif community [58, 101, 85], we assume meaningful higher-order dependencies to correspond to subgraphs of the true network, *true subgraphs*. However, unlike the network motif finding problem, we need not solve the subgraph-isomorphism problem for which no efficient solution is known. The subgraphs and the target graphs that we compare are labeled requiring us to solve a restricted version of the subgraph-isomorphism problem: matching the edge connectivity between the vertices from the true and inferred subgraphs.

We define two types of subgraph scores: Edge set match (E-score) and Vertex set match (V-score). E-scores measure how well *edges* of the true subgraphs are captured in the inferred network. V-scores measure how well *vertex degrees* of the true subgraphs are captured in the inferred network. While E-scores measure the overall ability of algorithms to capture correct subsets of edges, V-scores penalize algorithms that predict edges that are not part of the subgraphs. We enumerate different types of subgraphs such as cycles of length r (r-C), shortest-path reachable neighborhood (SPN) and r-radius neighborhood (r-N).

These subgraph scores measure how well subgraphs in the inferred network are matched to those in the true network. Both E and V-scores have a precision and recall definition. We use a “weight” per score, proportional to the subgraph size, which gives more importance to higher-order structures that are generally harder to match.

The E-score recall is:

$$R_{ES} = \sum_{g \in \mathcal{S}_t} \bar{w}_g \frac{|E_g \cap \widehat{E_g}|}{|E_g|}, \tag{3.1}$$

where $\mathcal{S}_t$ denotes a set of subgraphs of type t (E.g. $t =$ SPN for shortest path, $t = r$-RDN for r-radius neighbourhood, $t = r$-C for cycles), $\bar{w}_g = \frac{|V_g|}{\sum_{f \in \mathcal{S}_t} |V_f|}$ and V_g is the vertex set of subgraph g. E_g is the edgeset of g, and $\widehat{E_g}$ is the inferred edge set among $v \in V_g$. The E-score precision is defined similarly, except, E_g in the denominator is replaced by $\widehat{E_g}$.

The V-score recall is: $R_{VS} = \sum_{g \in \mathcal{S}_t} \bar{w}_g \frac{\sum_{v \in V_g} h_v}{\sum_{v \in V_g} h_v + m_v}$, where h_v and m_v are the number of neighbours of v, matched and missed, respectively, between the true and inferred networks. The V-score precision is defined similarly except m_v is replaced by $\widehat{m_v}$, the number of false neighbours of v.

These scores lie in the interval $[0, 1]$, and equal 1 when there is perfect match for all subgraphs. The above described scores are motivated by the walk-kernel used in the graph-kernel literature [114, 45]. In the walk-kernel the inner-product between two graphs is computed by a score very similar to Eq 3.1 but the fraction is replaced by 1 or 0 depending upon whether g exists in the two graphs.

3.2.2 Structural comparison of multiple networks

The above section assumes that we have one true and inferred network. That is we have data from only one condition. However for comparing networks across multiple conditions we need to adapt our structural comparison scores. Assume we have c true networks, one for each condition. Condition-specific network learning approaches have to learn c networks for all the conditions. An additional problem arises when we do not know the condition variable, and therefore do not which inferred network corresponds to which the condition. Our network structure comparison for multiple networks compares each true network with all of the inferred networks. We describe how we do this in practice using the 1-N subgraphs. Assume we are focusing on the i^{th} node in the j^{th} network, $1 \leq j \leq c$. Let $\mathbf{E}_{ij}$ be the set of edges connected to this node in its radius 1 neighborhood. The F-score F_{ESij} of matching $\mathbf{E}_{ij}$ is defined as:

$$F_{ESij} = \max_{j'} F_{ESij'}, \tag{3.2}$$

where $F_{ESij'}$ is the F-score from j'^{th} inferred network. The F-score itself is a harmonic mean of precision ($P_{ESij'}$) and recall ($P_{ESij'}$). The recall $R_{ESij'}$ is defined as

$$R_{ESij'} = \frac{|\mathbf{E}_{ij'} \cap \mathbf{E}_{ij}|}{|\mathbf{E}_{ij}|}, \tag{3.3}$$

where $\mathbf{E}_{ij'}$ is the edge set of i's 1-N subgraph in the j'^{th} inferred network. The precision is defined similarly except the denominator is $|\mathbf{E}_{ij'}|$.

By computing match scores for each node in the graph, we perform a more localized structure comparison by comparing the neighborhood of individual nodes. This allows us to compare algorithm A versus B based on the number of nodes on which A had significantly higher per-node neighborhood F-scores than B. In our comparisons of different algorithms for condition-specific network inference in Chapter 6, this is the method we use.

3.3 Functional comparison of inferred networks

The graphical models that we use to represent biological networks are composed of two parts: the graph structure and the potential functions describing the mathematical forms of the conditional distributions. The structural comparison described in the above sections only match the graph structure part. Comparison of the models also requires us to account for the functional part of the model. We now describe two scores of functionally comparing undirected graphical models.

3.3.1 Pseudo likelihood

Pseudo likelihood based model comparison is done by computing the pseudo likelihood of a hold-out test set given a model [64, 105]. This is similar in flavor to model comparison based on data likelihood [65]. The pseudo likelihood of a graph over random variable set

$\mathbf{X}$ with parameters $\theta = \{\theta_1, \cdots, \theta_N\}$ is an approximation to likelihood of data given an undirected graphical model:

$$P(\mathbf{X} = \mathbf{x}) = \prod_{i=1}^{N} P(X_i = x_i | \mathbf{M}_i = \mathbf{m}_i) \tag{3.4}$$

where $\mathbf{M}_i$ denotes the Markov blanket of X_i, $\mathbf{x}$ is an assignment to $\mathbf{X}$. Log of pseudo likelihood gives us a decomposable score for a graph:

$$PLL(\mathcal{D}; \theta) = \sum_{d=1}^{\mathcal{D}} \sum_{i=1}^{N} \log P(X_i = x_{di} | \mathbf{M}_i = \mathbf{m}_{di}, \theta_i) \tag{3.5}$$

Here $\mathcal{D}$ is the hold out test set. To compare two learned models from two different methods, we compare again perform a localized comparison on a per-variable basis. In particular, we compare each variable's contribution to the overall pseudo log likelihood. Let $\text{PLLV}(X_i, \mathbf{M}_i)$ denote the contribution of X_i to the overall pseudo log likelihood, that is $\text{PLLV}(X_i, \mathbf{M}_i) = \sum_{d=1}^{|\mathcal{D}|} = P(X_i = x_{di} | \mathbf{M}_i = \mathbf{m}_{di})$. For each method, we learn different models on different folds of the training data, compute the $\text{PLLV}(X_i, \mathbf{M}_i)$ for each fold, and then compare the average $\text{PLLV}(X_i, \mathbf{M}_i)$ across folds. A method is said to be better if it has more variables with significantly higher pseudo log likelihood than the other method.

3.3.2 Symmetric Kullback-Leibler (KL) divergence

Symmetric KL divergence provides another method of evaluating different learned models. Symmetric KL divergences can be used as a measure of distance between the distributions specified by the true and the inferred models. Let p and q denote two probability distributions. The KL divergence between these two distributions is

$$\text{KL}(p, q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

This is usually asymmetric, that is, $\mathrm{KL}(p, q) \neq \mathrm{KL}(q, p)$. Symmetric KL divergence is defined as

$$\mathrm{SymmKL}(p, q) = \mathrm{KL}(p, q) + \mathrm{KL}(q, p)$$

The integral can be computed in closed form only for special forms for p and q, such as Gaussians. For the majority of the cases, we need to numerically estimate the integral using a large hold out test set $\mathcal{D}$:

$$\mathrm{SymmKL}(p, q) = \sum_{d=1}^{\mathcal{D}} p(X_i = x_{di})\log\frac{p(X_i = x_{di})}{q(X_i = x_{di})} + q(X_i = x_{di})\log\frac{q(X_i = x_i)}{p(X_i = x_i)}$$

Here p is the conditional probability distribution of X_i given its Markov blanket in the true model, and q is the conditional distribution of X_i in the inferred model. Similar to the pseudo log likelihood case, we compute symmetric KL divergence for each variable per method, over different folds of the training data and compare the average symmetric KL divergence of the two methods. Unlike pseudo log likelihood, a method is better than another if it has more variables with significantly lower symmetric KL divergence. This is because symmetric divergence measures the distance between the true and inferred models and a good inferred model must be as close in distance as possible to the true model.

3.4 Biological validation

Biological validation is the process of validating inferred networks using existing literature, and controlled vocabulary based annotation of genes [6]. We applied several novel and existing measures of biological validation of our inferred networks. We defined the *Annotation-Topological Similarity*, a novel global measure of assessing the quality of the inferred network structure, based on the agreement between annotation and topological similarity of two subgraphs. The existing validation methods are based Gene Ontology

[6], a controlled vocabulary for annotating genes, and semantic similarity of genes connected in the inferred network [89]. We describe these validations below.

3.4.1 Annotation topological measure

We developed a global measure of validation of the entire graph structure, Annotation-Topological Similarity (ATS), which assesses if subgraphs that are topologically close in the inferred network, are also similarly annotated. The Annotation Topological similarity computes the correlation between annotation and topological similarity for pairs of subgraphs.

Annotation similarity

Annotation similarity, a_{ij} between two subgraphs S_i, and S_j measures the similarity between the subgraphs in annotation space. For each S_i, we compute an enrichment p-value using the hypergeometric distribution for each of the 33 Gene Ontology Slim (GOSlim) process terms. Thus in annotation space, each subgraph is represented by a 33 dimensional vector, A_i, where the k^{th} element, $A_i(k)$, is the logarithm of the p-value of enrichment in the k^{th} term using genes from S_i. The annotation similarity, a_{ij} between S_i and S_j, is given by the Pearson's correlation value between their respective annotation enrichment vectors A_i and A_j.

Topological similarity

The topological similarity, t_{ij} between two subgraphs S_i and S_j captures how close these subgraphs are on the complete graph. We use two definitions of topological similarity. The first is based on the average shortest path distance between the vertices of S_i to vertices of S_j:

$$t_{ij} = \frac{\sum_{v \in V_i, u \in V_j} 2 * d(u,v)}{|V_i| *| V_j|}, \tag{3.6}$$

where V_i and V_j are the vertex sets of S_i and S_j respectively.

The second definition is obtained from Lee *et al.* [83], and is based on the number of edges that cross between vertices of S_i and S_j:

$$t_{ij} = \frac{l_{ij}}{n_i + n_j} \tag{3.7}$$

where l_{ij} is the sum of number of nodes common between S_i and S_j, and the number of edges across S_i and S_j. n_i and n_j are the vertex counts in subgraphs, S_i and S_j. Subgraphs that are not connected in the graph are not considered for computing the ATS measure.

Finally, the ATS measure is the Pearson's correlation between the annotation and topological similarity over all pairs of subgraphs. If our measure of topological similarity is the shortest path distance, we expect the topological distance and annotation similarity to be anti-correlated, producing a negative correlation coefficient. Therefore the more negative the coefficient, i.e. small values of t_{ij}, the more likely it is for two subgraphs with similar function to be close together, and two subgraphs with different function to be far apart. If our measure of topological similarity is from Lee *et al.*, we expect the topological distance and annotation similarity to be correlated, producing a positive correlation coefficient, and the more positive the coefficient, the better the quality of the inferred dependencies.

In our application of the ATS score, we considered subgraphs generated from each vertex and its neighbors that are 1 step away. These subgraphs had to be maximal, that is no vertex set of a subgraph could be completely contained within another vertex set. ATS was the Pearson's correlation coefficient between two vectors, v_A and v_T, each of length $\binom{|\mathbf{S}|}{2}$, where S is the set of subgraphs generated from the inferred network. Each dimension, $v_A(r)$ is the annotation similarity for r^{th} subgraph pair, where $1 \leq r \leq \binom{|\mathbf{S}|}{2}$. Similarly, each dimension, $v_T(r)$ is the topological distance for r^{th} subgraph pair. We now define annotation and topological distance.

We also evaluated the statistical significance of the ATS measure per inferred network using a background distribution of ATS from random networks. Specifically, for each inferred network, we generated 100 random networks using a randomization technique that preserved the degree distribution of the networks [101], computed ATS for each random network, and generated the histogram of the ATS values.

3.4.2 Gene ontology enrichment analysis

Gene ontology (GO) is a hierarchically organized, controlled vocabulary used to annotate the genes of an organism [6]. Gene ontology has three types of terms: (a) biological process, (b) cellular location, and (c) molecular function. We assess biological importance of an inferred subgraph using a p-value enrichment of GO term. The p-value tells the probability of observing by random chance h_i out of g_i genes in the i^{th} subgraph associated with a term t, given that there are n_a out of n_b genes associated with t. We use the hypergeometric distribution for computing this p-value.

We use false-discovery rate (FDR) to avoid the problem of multiple hypothesis correction. FDR is estimated using a simulation strategy similar to Boyle *et al.* [20]. Briefly, assume we sample $s = 1000$ random subsets of size k from the entire gene set. We compute their p-values for each GO term. The FDR is associated with the number of terms enriched in a subgraph at a particular p-value. For example, if a subgraph of size k is enriched in u terms at $p < 10^{-4}$, FDR is $\frac{u'}{u}$, where u' is the average number of terms enriched in a random subgraph of size k at $p < 10^{-4}$. We used an FDR of ≤ 0.05 to identify significant enrichments.

3.4.3 Semantic similarity

We use the definition of semantic similarity from Lord *et al.* [89]. Semantic similarity is defined between two annotation terms using the maximal amount of information present in a common ancestor of the terms. For GO terms the information is inversely proportional to the number of genes that are annotated with a term. Hence, a very specific term with few genes has more information than a broader term that has many more genes annotated with it. The semantic similarity of two genes is defined using the semantic similarity of sets of terms associated with the two genes. Let g_i and g_j be two genes connected by an edge in our inferred network. Let $\mathbf{T}_i$ and $\mathbf{T}_j$ be the set of GO process terms with which the genes g_i and g_j are annotated, respectively. We compute an average semantic similarity, simg_i, g_j for all pairs of terms as:

$$\text{sim}(g_i, g_j) = \frac{1}{|\mathbf{T}_p| *| \mathbf{T}_q|} \sum_{t_p \in \mathbf{T}_i, t_q \in \mathbf{T}_j} \text{semsim}(t_p, t_q),$$

where $\text{semsim}(t_p, t_q) = -\log(\min_{a \in \mathbf{P}_{pq}} p_a)$, and $\mathbf{P}_{pq}$ is the set of common ancestors of the terms t_p and t_q in the GO process "is-a" hierarchy. p_a is the probability of the term, and is defined as the ratio of the number of genes annotated with the term a to the total number of genes with a GO process assignment, and $-\log(p_a)$ is the amount of information associated with a term a.

Chapter 4

Learning undirected graphical models for biological networks

In this chapter, we investigate the ability of undirected probabilistic graphical models to represent biological networks. The work described in this chapter is necessary to (a) justify the class of models that we will use to represent condition-specific networks, (b) pin-down modeling questions about the probability distributions used to mathematically represent the statistical dependencies. We describe two novel algorithms for learning the structure of the undirected models. The first algorithm is based on finding consistent Markov blanket structures and the second algorithm describes a scalable framework for learning these networks. We then present results using simulated data and discuss issues related to the representation of the conditional probability distributions.

4.1 Representing biological networks as undirected probabilistic graphs

Probabilistic graphical models (PGMs) representing real-world networks capture important structural and functional aspects of the network by describing a joint probability distribution of all node measurements. The structure encodes conditional independence assumptions allowing the joint probability distribution to be tractably computed. When the structure is unknown, likelihood-based structure learning algorithms are employed to infer the structure from observed data.

Likelihood-based structure learning of directed acyclic graphs (DAGs), such as Bayesian networks, is widely used because the likelihood score can be tractably computed for all candidate DAGs. However, in many domains such as biology, causal implication of directed edges is difficult to ascertain without perturbations, leaving only a correlation implication. In such situations, undirected graphical models are a more natural representation of statistical dependencies. Unfortunately, likelihood-based structure learning of these models is much harder due to the intractability of the partition function [1].

To overcome this issue, researchers have opted several alternatives: learn graphical Gaussian models where the likelihood can be computed tractably [87]; restrict to lower order, often pairwise functions, [91, 84]; use pseudo-likelihood as structure score instead of likelihood [15]; learn dependency networks [64, 121]; or learn Markov blanket canonical factors [1]. Pairwise models are scalable, but, approximate higher-order dependencies by pairwise functions, which is limiting for domains where higher-order dependencies occur commonly. While dependency networks are scalable, each variable neighborhood is estimated independently, resulting in inconsistent structures when the data sample size is small. This is problematic for real-world data which often lack sufficient samples to guarantee a consistent joint probability distribution for the learned structure. Finally, Markov blanket canonical parameterization requires exhaustive enumeration of variable subsets up to a pre-specified size l, which is not scalable for networks with hundreds of nodes.

We have developed a new algorithm for learning undirected graphical models, that produces consistent structures and is scalable to be applicable for real-world domains. Our algorithm, Markov blanket search (MBS) is inspired by Abbeel *et al.*'s Markov blanket canonical parameterization, which established an equivalence between global canonical potentials and local Markov blanket canonical factors (MBCFs) [1]. We extend Abbeel *et al.*'s result to establish further equivalence between MBCFs and *per-variable canonical factors*. Because per-variable canonical factors require learning Markov blankets *per-variable*, rather than all subsets up to size l, we save $O(n^{l-1})$ computations during structure learning, where n is the number of variables. The equivalence of per-variable canonical factors and global canonical factors has been observed before [107]. However, we are the first to use per-variable canonical factors in the context of MRF structure learning to learn consistent MRF structures. Enforcing structural consistency during search, guarantees the structure to be a MRF, and also the existence of a joint distribution for the individual conditional distributions. Thus we need not perform additional post-processing to guarantee consistent structures [121].

4.1.1 Markov random fields

A Markov random field (MRF) is an undirected, probabilistic graphical model that represents statistical dependencies among a set of random variables (RVs), $\mathbf{X} = \{X_1, \cdots, X_n\}$. A MRF consists of a graph $\mathcal{G}$ and a set of potential functions $\psi = \{\psi_1, \cdots, \psi_m\}$, one for each clique in $\mathcal{G}$. The graph structure describes the statistical dependencies, and the potentials describe the functional relationships between the RVs. The RVs encode the observed measurements for each node, $X_i \in \mathcal{R}$. The joint probability distribution of the MRF is defined to be: $P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z}\prod_{i=1}^{m}\psi_i(\mathbf{F}_i = \mathbf{f}_i)$, where $\mathbf{x}$ is a joint assignment to $\mathbf{X}$, $\mathbf{F}_i \subseteq \mathbf{X}$ is the variable set in the i^{th} clique, associated with ψ_i; $\mathbf{f_i} \subseteq \mathbf{x}$ is a joint assignment to $\mathbf{F}_i$. Z is the partition function and is defined as a summation over all possible joint assignments of $\mathbf{X}$.

Structure learning of MRFs using likelihood is difficult in general because of Z [1]. This is because estimating Z requires a sum of exponentially many joint configurations of the RVs, making it intractable for real-world domains. To overcome this problem, researchers have proposed approaches that use pseudolikelihood [15, 64], or, have used Markov blanket canonical parameterization (MBCP) [1]. We use an approach similar to MBCP, which requires the estimation of optimal Markov blankets for RV subsets, $\mathbf{Y} \subseteq \mathbf{X}$, $|\mathbf{Y}| \leq l$, where l is a pre-specified, maximum subset size. However, we learn local per-variable factors, requiring estimation of Markov blankets of only individual RVs. Avoiding Markov blanket estimation of all subsets, makes our approach scalable to domains with hundreds of nodes. We give details of the proof in the Appendix A and we now proceed with the algorithm description.

4.1.2 Markov blanket search (MBS) algorithm

The MBS algorithm learns the structure of a MRF by finding the best neighborhood or Markov blanket (MB) for each RV. To identify the best MB, we need to optimize a *score*, $S(X_i|\mathbf{M}_i)$ per RV X_i, which quantifies dependence between a RV and its MB. Examples of such scores include pseudolikelihood (dependency networks) or conditional entropy (MBCP) [32]. For example, the best MB identified via conditional entropy, $H(X_i|\mathbf{M}_i)$ is: $\mathbf{M}_i = \arg\min_{\widehat{\mathbf{M}}_i} H(X_i|\widehat{\mathbf{M}}_i)$. Best MB via pseudolikelihood, $PLL(X_i|\mathbf{M}_i)$, is: $\mathbf{M}_i = \arg\max_{\widehat{\mathbf{M}}_i} PLL(X_i|\widehat{\mathbf{M}}_i)$

Dependency networks and MBCP identify the best MB per RV by independently optimizing $S(X_i|\mathbf{M}_i)$ per RV[1]. However, optimizing $S(X_i|\mathbf{M}_i)$ per RV independently, may

[1] In MBCP estimation, MBs of variable sets are identified independently. MBCP requires an additional subset consistency check: if $\mathbf{X} \subset \mathbf{Y}$, then $\mathbf{M}_{\mathbf{X}} \subset (\mathbf{M}_{\mathbf{Y}} \cup (\mathbf{Y} \setminus \mathbf{X}))$

Algorithm 1 Markov Blanket Search

```
Input:
  Random variable set, $\mathbf{X} = \{X_1, \cdots, X_{|\mathbf{X}|}\}$
  maximum neighborhood sizes, $k_{max}$, $k_{hard}$
Output:
   Inferred graph structure $\mathcal{G}$
for $k = 1; k \leq k_{max}; k++$ do
  for $X_i \in \mathbf{X}$ do {Add stage}
    Find best new MB variable $X_j$ that maximizes $\Delta S_{ij}$ s.t. $|\mathbf{M}_i| \leq k$ (Eq 4.1)
  end for
  for  $X_i \in \mathbf{X}$ do {Swap stage}
    for $X_j \in \widehat{\mathbf{M}_i}^k$ do
      for $X_q \in \mathbf{X} \setminus (\widehat{\mathbf{M}_i}^k \cup \{X_i\})$ and $|\widehat{\mathbf{M}_q}^k| \leq k_{hard}$ and $X_q \notin$ tabulist$(X_i)$ do
        Delete $\{X_i, X_j\}$, add $\{X_i, X_q\}$, add $X_j$ to tabulist$(X_i)$ if swapping $X_q$ for $X_j$ gives
        maximal score improvement.
      end for
    end for
  end for
end for
```

result in inconsistent MBs. In particular, we cannot guarantee that if $X_j \in \mathbf{M}_i$, then $X_i \in \mathbf{M}_j$. This inconsistency can be handled as a post-processing of the learned MBs [121]. However, our experiments suggest that a post-processing approach produces lower quality MBs (Section 4.1.4).

We propose a different approach that finds consistent MBs during the search process. To find consistent MBs, we search MBs, not only using the improvement in $S(X_i|\mathbf{M}_i)$ on adding X_j, but also the score change in $S(X_j|\mathbf{M}_j)$ if X_i was added to $\mathbf{M}_j$. This is done by computing the net score gain per candidate MB for X_i. Let $\widehat{\mathbf{M}_i}^{k-1}$ denote the best MB for X_i obtained so far. Then the score gain is:

$$\begin{aligned}\Delta S_{ij} = {} & S(X_i|\widehat{\mathbf{M}}_i^{k-1}) - S(X_i|\widehat{\mathbf{M}}_i^{k-1} \cup \{X_j\}) \\ & + S(X_j|\widehat{\mathbf{M}}_j^{k-1}) - S(X_j|\widehat{\mathbf{M}}_j^{k-1} \cup \{X_i\}) \end{aligned} \tag{4.1}$$

Our approach is similar to Hofmann & Tresp's edge-based score for guaranteeing consistency [66]. However, their search strategy starts from a fully connected network and removes edges, whereas we add and replace edges starting with a completely disconnected network. For real-world domains, growing larger neighborhoods from smaller neighborhoods is more feasible than shrinking large neighborhoods, because we may not have enough data for reliably learning large neighborhoods.

The MBS structure learning algorithm uses Eq 4.1 to greedily identify the best MB for each variable (Algo. 1). Each iteration uses a combination of *add* and *swap* operations to learn the best structure. In the add stage of the k^{th} iteration, we make one variable extensions to the current $\widehat{\mathbf{M}}_i^{k-1}$ of each X_i, restricting to at most $k \leq k_{max}$ RVs per MB. Instead of adding one edge at a time, we add all possible edges that could be added to the current graph G_k. Thus, we first propose candidate extensions for each variable given the current graph G_k, and then make edge additions in order of decreasing score improvement.

In the swap stage, we revisit all variables $X_j \in \widehat{\mathbf{M}}_i^{k}$ for each X_i, and consider other RVs, $X_q \notin (\{X_i\} \cup \widehat{\mathbf{M}}_i^{k})$, which if swapped in instead of X_j, gives a score improvement. If so, we replace X_j by X_q with the maximal score improvement, in $\widehat{\mathbf{M}}_i^{k}$, and store X_j in the *tabu list* of X_i. This prevents X_j from being included in X_i's MB in subsequent iterations. In the swap stage, a variable can be present in $> k_{max}$ MBs. However, no variable can be in more than $k_{hard} = 20$ MBs. Thus, nodes in our inferred networks have degrees $\geq k_{hard}$, which reasonably models hub nodes in most domains.

The per-variable canonical factor equivalence exploited by MBS to identify the MRF structure does not make any specific assumptions of the parametric form of the conditional probability distributions. MBS only requires that the candidate MBs be scored using the conditional probability distributions. So MBS can potentially be instantiated with any probability distribution and choice of score. For empirical evaluation of our framework, we selected $P(X_i|\mathbf{M}_i)$ to be conditional Gaussians and $S(X_i|\mathbf{M}_i)$ to be the regularized conditional entropy for each X_i: $S(X_i|\mathbf{M}_i) = H(X_i|\mathbf{M}_i) + \lambda\text{log}(|\mathbf{M}_i|)$. $\lambda\text{log}(|\mathbf{M}_i|)$ penalizes large MBs and $0 \leq \lambda \leq 1$ is a regularization coefficient. We discuss in the next section how different types of conditional distributions can be used.

4.1.3 Family of potential functions

We implemented the MBS algorithm with three types of functions for representing potential distributions. These are regression trees, kernel density estimators and multivariate Gaussians. In all these cases we need to be able to compute the joint entropy, because from the joint entropy the conditional entropy $H(X_i|\mathbf{M}_i)$ can be easily computed as $H(X_i|\mathbf{M}_i) = H(X_i, \mathbf{M}_i) + H(\mathbf{M}_i)$.

Multi-variate Gaussians

Information-theoretic measures such as entropy can be computed in closed form for multivariate Gaussians. Specifically from Cover and Thomas [32], the joint entropy of $\{X_i\} \cup \mathbf{M}_i$ is given by

$$H(X_i, \mathbf{M}_i) = \frac{1}{2}\left(l(1 + \text{log}2\pi) + |\Sigma_i|\right) \tag{4.2}$$

where $l = |\mathbf{M}_i| + 1$, Σ_i is the covariance matrix for the multivariate Gaussian describing the joint distribution of $\{X_i\} \cup \mathbf{M}_i$. The conditional entropy can be easily computed from joint entropy as $H(X_i|\mathbf{M}_i) = H(X_i, \mathbf{M}_i) - \sum_{Y \in \{X_i\} \cup \mathbf{M}_i} H(Y)$.

Kernel density estimators

Kernel density estimators (KDE) are non-parameteric families of probability distribution. We first define KDE estimators for univariate probability distribution of a random variable X. The KDE estimates the probability of X taking value x as a sum, $P(X = x$:

$$P(X = x) = \frac{1}{nh}\sum_{j=1}^{n} K(\frac{x - x_j}{h}) \tag{4.3}$$

where K is a kernel function, h is the width of the kernel and n is the number of data points our data set. h determines the smoothness of the estimation. Well known examples of the kernel function are Gaussian kernels, exponential kernels or the simplest histogram kernels. Entropy estimation for probability distributions represented by KDE cannot be computed in closed form and several approximations have been proposed [11]. We estimate the marginal entropy, $H(X)$ of X as the following:

$$H(X) = -\frac{1}{n}\sum_{j=1}^{n} logP(X = x) \tag{4.4}$$

where $P(X = x)$ is obtained from the KDE estimator. The entropy can be considered as an expectation of X, and the above estimate computes the expected value with respect to the uniform distribution.

However, in our setup we need KDE estimators for multi-variate distributions. To accomplish this efficiently, we use the Figtree library [102]. The joint entropy of a set of a set of random variables $\mathbf{X}$ is given by a similar formula:

$$H(\mathbf{X}) = -\frac{1}{n}\sum_{j=1}^{n} logP(\mathbf{X} = \mathbf{x}) \tag{4.5}$$

An important issue in KDE is selecting h. We use a hold-out set of estimating h in our experiments.

Regression trees

Regression trees are another non-parametric family for representing the probability distribution associated with each family [64, 125]. Regression trees allow us to directly represent the conditional distribution, $P(X_i|\mathbf{M}_i)$. Each non-leaf node of the regression tree represents a split on one of the variables $X_i \in \mathbf{M}_i$. The conditional probability $P(X_i = x_i|\mathbf{M}_i = \mathbf{m}_i)$ is given by tracing the path down the regression tree based on the joint assignment $\mathbf{m}_i$. At each leaf node l we have a univariate Gaussian with mean μ_{il} and variance σ_{il}, which are estimated using the values of X_i that fall into the node l. For the regression tree we compute the conditional entropy directly, as opposed to the multi-variate Gaussian and KDE. The conditional entropy, $H(X_i|\mathbf{M}_i)$ is given by the formula:

$$H(X_i|\mathbf{M}_i) = -\int P(\mathbf{M}_i)\text{log}P(X_i|\mathbf{M}_i) \tag{4.6}$$

Again, this integral cannot be computed in closed form, but we approximate it by summing over all the leaf nodes. This is because the structure of the regression tree associated with this conditional distribution, partitions assignments to $\mathbf{M}_i$ into the sets, where each set is consistent with one path of the regression tree leading to a leaf node. Therefore the probability $P(\mathbf{M}_i)$ is the normalized count of the number of assignments to $\mathbf{M}_i$ that are consistent with a path, which in turn is the number of data points residing in the leaf node. Thus the conditional entropy using the regression tree is

$$H(X_i|\mathbf{M}_i) = -\sum_{l=1}^{|\mathcal{L}_i|} \frac{n_l}{\sum_o n_o} \frac{1+\log(2\pi\sigma_{il})}{2} \tag{4.7}$$

The last term is the marginal entropy of X_i using the data points assigned to the l^{th} leaf node. An issue in the regression tree is that of controlling the size of the tree. Very large trees can describe the data well but can result in to over-fitting the data. We treat p as an additional regularization term and learn networks for a wide range of p. Although we did have the regularization of the size of the Markov blanket, λ, its effect was minimal compared to the p. Hence, in our experiments we fixed the value of λ.

4.1.4 Results

The goal of our experiments was (a) to identify the most appropriate form of conditional probability distributions for the MBS algorithm, (b) compare MBS against other existing algorithms for learning both directed and undirected graphical models, (c) demonstrate the importance of incorporating Markov blanket consistency during network structure learning.

Dataset description

All experiments described in this section were done on simulated data from networks of known topology enabling a direct validation of the inferred structures. The datasets were generated by a gene regulatory network simulator using differential equations for describing gene and protein expression dynamics [115]. The simulator models combinatorial control among regulator proteins to generate expression data resembling data from real-world networks. We generated six datasets using three networks of different sizes (Table 4.1). Each sample consists of steady-state expressions reached after perturbing the kinetic constants of the genes. Networks for G50 and G75 were generated *de novo* by the simulator. The network for ECOLI belongs to the bacteria, *E. coli*.

As the true network topologies for these data are known, our comparisons were based on the match between the inferred and true network structures. We used various scores described in Chapter 3 to compare the true and inferred graphs. Briefly, we extracted subgraphs of different types (e.g. cycles, neighborhood) from the true network and used an F-score measure to match the vertex neighborhood and edge set per subgraph. We refer to the scores for vertex neighborhood as *V-scores* and for edge set as *E-scores*. We use shortest path neighborhoods (SPN), r-radius neighborhoods comprising a vertex and its neighbors $\leq r$ steps away ($r \in \{1, 2\}$, denoted by 1N and 2N), and cycles of size r ($r \in \{3, 4\}$, denoted by 3C and 4C). ECOLI datasets did not have any cycles.

Network	Number of Nodes	Datasets
G50	100	G50-ALL, G50-TF
G75	150	G75-ALL, G75-TF
ECOLI	188	ECOLI-ALL, ECOLI-TF

Table 4.1: Description of simulated datasets generated from RENCO. The ALL datasets were generated by perturbing kinetic constants of all genes, and the TF datasets were generated by perturbing kinetic constants of only the regulator genes.

Effect of different potentials

We performed preliminary experiments to evaluate the performance of the MBS algorithm with different types of potential functions for the conditional distribution (Fig 4.1). These experiments were performed on MBS without the swap stage. The kernel width, h for the KDE was estimated on a hold-out set which optimized the MBS score by varying the h in the range $1e - 10 \leq h \leq 1$, changing h by a factor of 5 at each step. The leaf node size, l of the Regression tree was varied in $l \in \{10, 30, 50, 70\}$.

We found MBS with multi-variate Gaussian to have the best performance on the E scores. The regression tree was occasionally better than the KDE model especially for the TF datasets. On the V scores, the multi-variate Gaussian was outperformed for a few cases (G50-ALL, neighborhood scores), but in general was at par or better than the MBS versions using other probability distribution forms. The regression tree was closer to the multi-variate Gaussian than the KDE. In summary, we found the multi-variate Gaussians to have the best performance. The experiments described in the following sections were done using MBS with multi-variate Gaussians.

Comparison of MBS against other algorithms

We compared our Markov Blanket Search (MBS) algorithm against existing algorithms for undirected and directed graphs on simulated data described in Section 4.1.4. We compared

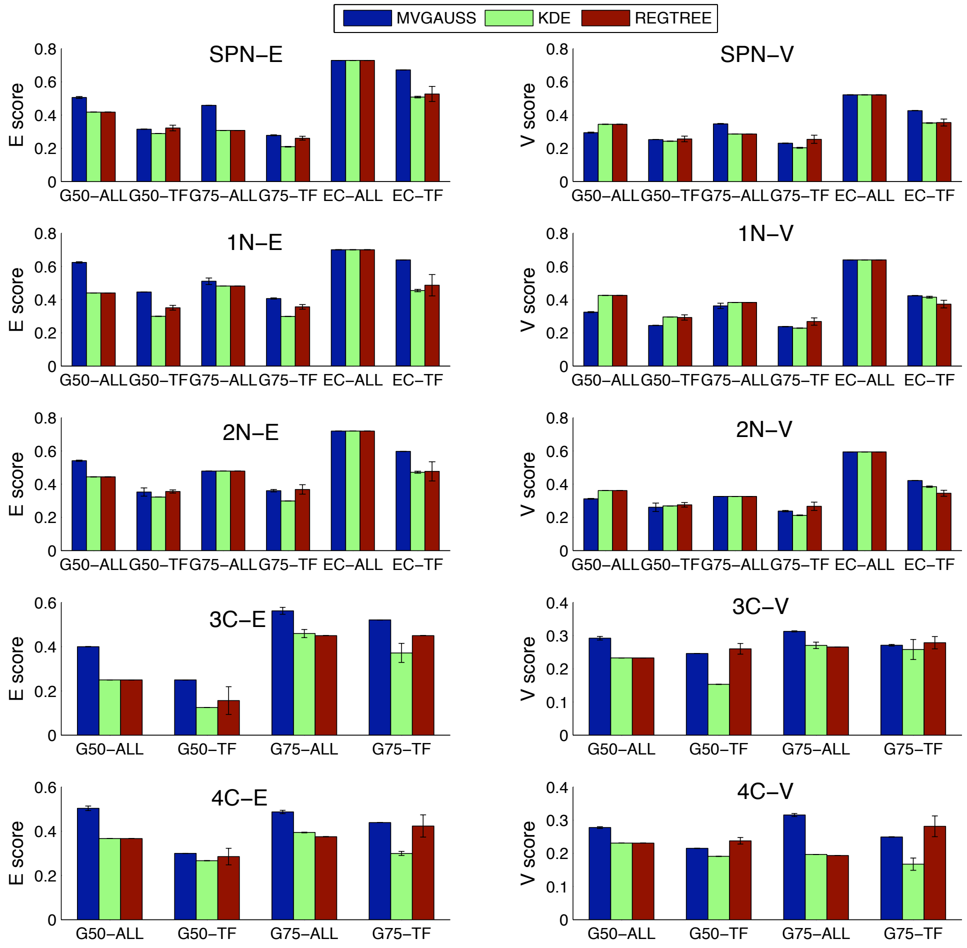


Figure 4.1: Performance comparison of different models for the conditional distribution. The x-axis is for datasets.

MBS to two undirected algorithms: ARACNE [91], and a Lasso regression-based Graphical Gaussian model (GGLAS) [87]. We also compared MBS against several directed models provided in the DAGLearn software[2]: full DAG search (FULLDAG), LARs based order search (ORDLAS), DAG search using Sparse candidate for pruning (SPCAND), and DAG search using L1 regularization based Markov blanket estimation (L1MB) [121]. Because L1MB does not learn consistent Markov blankets, a post-processing step is required to make the structures consistent. The AND post-processing removes X_i from M_j if

[2] `http://www.cs.ubc.ca/~murphyk/Software/DAGlearn/`

$X_j \notin \mathbf{M}_i$, where $\mathbf{M}_i$ and $\mathbf{M}_j$ are X_i's and X_j's MB, respectively. The OR post-processing includes X_i in $\mathbf{M}_j$ if $X_j \in \mathbf{M}_i$. We refer to L1MB with AND and OR post-processing as MBAND and MBOR, respectively. We also included an implementation of order MCMC (ORDMC) for Bayes net search (`http://www.bioss.ac.uk/staff/.adriano/comparison/comparison.html`).

Our comparison used E and V-scores averaged over four runs per algorithm corresponding to different settings of an algorithm-specific parameter. This parameter is λ in MBS (Section 4.1.2), data processing inequality d in ARACNE, and hyper-prior parameter for the variance in GGLAS. For all DAG searches other than ORDMC, we used different random restart probabilities to generate different candidate graphs. In our experiments, $\lambda \in \{1e-5, 3e-5, 5e-5, 7e-5\}$ and $d \in \{0, 0.3, 0.5, 0.7\}$. All simulated experiments used $1 \leq k_{max} \leq 11$. For ORDMC, we varied the edge posterior probability. For each parameter setting, the graph with the highest average of E and V score is used. We compare the best graph per algorithm across different parameter settings.

Our complete results are summarized in Table 4.2. For all datasets other than ECOLI-ALL, MBS significantly beats all algorithms at least as often as it is beaten (Student's t-test, p-value ≤ 0.05). On ECOLI-ALL, ARACNE outperforms all algorithms, suggesting that ECOLI-ALL likely does not contain many higher-order dependencies. There is no significant difference between MBS and ARACNE on ECOLI-TF, which is generated from the same network as ECOLI-TF using different perturbations.

We find that the performance margin between MBS and the DAG learning models is greater than undirected learning algorithms, suggesting undirected graphs may be better representations for this domain. Overall, MBS does a better job of learning the network structures compared to both directed and undirected algorithms for majority of the datasets. Although the score improvements are marginal, they are statistically significant.

			MBS	ARACNE	ORDMC	FULLDAG	ORDLAS	SPC	MBAND	MBOR	GLASSO
G50-ALL	E	SPN	0.550	**0.418**	**0.415**	0.535	**0.492**	0.446	**0.416**	**0.406**	**0.463**
		1N	0.661	**0.44**	0.552	0.621	**0.636**	0.456	**0.446**	**0.428**	**0.836**
		2N	0.589	**0.444**	**0.44**	**0.519**	**0.532**	**0.510**	**0.534**	**0.438**	**0.562**
		3C	0.800	0.400	0.438	0.790	0.792	0.784	0.783	**0.250**	0.866
		4C	0.645	**0.440**	0.475	0.636	0.630	**0.580**	0.638	**0.367**	0.721
	V	SPN	0.308	**0.345**	0.326	**0.248**	**0.264**	**0.262**	0.292	**0.338**	**0.379**
		1N	0.352	**0.426**	0.330	0.284	**0.285**	**0.276**	0.348	**0.416**	0.35
		2N	0.335	**0.361**	0.344	**0.265**	**0.273**	**0.261**	323	**0.353**	0.356
		3C	0.328	**0.251**	**0.201**	0.301	**0.284**	**0.287**	**0.401**	**0.233**	0.259
		4C	0.324	**0.246**	**0.205**	**0.267**	**0.281**	**0.275**	**0.356**	**0.230**	0.248
G75-ALL	V	SPN	0.493	**0.363**	**0.31**	0.436	**0.424**	**0.455**	**0.423**	**0.304**	0.430
		1N	0.577	**0.475**	0.573	0.576	0.573	**0.521**	**0.535**	**0.478**	**0.811**
		2N	0.504	**0.444**	**0.457**	0.502	0.497	**0.475**	0.495	**0.469**	0.509
		3C	0.6	0.604	0.588	0.590	0.637	0.621	**0.529**	**0.45**	**0.824**
		4C	0.573	0.516	0.492	0.602	0.609	0.610	**0.506**	**0.375**	**0.755**
	E	SPN	0.367	0.377	**0.29**	**0.252**	**0.253**	**0.328**	**0.297**	**0.283**	0.262
		1N	0.417	0.439	0.299	**0.268**	**0.267**	**0.335**	**0.342**	**0.379**	**0.303**
		2N	0.377	0.381	**0.323**	**0.266**	**0.254**	0.322	**0.298**	**0.32**	**0.334**
		3C	0.345	0.339	**0.225**	**0.214**	**0.23**	0.313	**0.246**	**0.263**	**0.216**
		4C	0.306	0.351	**0.193**	**0.257**	0.272	0.354	**0.258**	**0.192**	**0.182**
ECOLI-ALL	E	SPN	0.747	0.753	**0.729**	0.718	0.703	0.759	0.758	**0.729**	**0.729**
		1N	0.751	0.776	0.700	0.725	0.690	**0.778**	0.748	**0.705**	**0.700**
		2N	0.726	0.752	0.719	0.701	0.679	**0.749**	**0.746**	0.724	0.719
	V	SPN	0.514	**0.567**	0.522	**0.271**	**0.303**	**0.354**	0.43	0.520	0.522
		1N	0.608	**0.667**	**0.639**	**0.289**	**0.326**	**0.396**	0.485	**0.627**	**0.639**
		2N	0.591	**0.622**	0.594	**0.272**	**0.308**	**0.376**	**0.454**	0.585	0.594
ECOLI-TF	E	SPN	0.68	0.592	**0.55**	**0.504**	**0.522**	**0.608**	**0.542**	**0.507**	**0.730**
		1N	0.655	0.56	0.591	**0.46**	**0.486**	**0.574**	**0.5**	**0.448**	**0.700**
		2N	0.618	0.532	0.48	**0.418**	**0.454**	**0.542**	**0.493**	**0.456**	**0.719**
	V	SPN	0.460	0.449	**0.359**	**0.194**	**0.183**	**0.307**	**0.293**	**0.369**	**0.523**
		1N	0.479	0.474	0.307	**0.196**	**0.179**	**0.289**	**0.301**	**0.401**	**0.639**
		2N	0.447	0.439	0.37 0	**0.182**	**0.164**	**0.266**	**0.282**	**0.376**	**0.594**

Table 4.2: Comparison of MBS algorithm against existing algorithms for directed and undirected graphs. Results from four datasets are shown. Rows give different structure scores; columns are different structure learning algorithms; each entry is an E or V score. **blue** and **red** indicate that MBS performs significantly **better** or **worse** than the algorithm compared. SPN: shortest path neighborhood, 1N, 2N:$r = 1$ and $r = 2$ neighborhood, 3C, 4C: cycles of size 3 or 4.

Structural consistency for pruning DAGs

To assess the value of enforcing consistency during learning, rather than as a post-processing step, we used the MBS-learned Markov blankets as family constraints in DAG search algorithms. We compared the DAG structures constrained using MBS Markov blankets against those constrained by Sparse candidate (SPCAND) and L1 MB regularization (L1MB). L1MB uses either an OR or AND of the Markov blankets to generate consistent Markov blankets.

We used the maximum size of L1MB AND and OR Markov blankets as the neighborhood size, k, for MBS and SPCAND. We first compared L1MB with OR post-processing (MBOR) using $k = 11$ for both G50-ALL and G75-ALL (Table 4.3). We found the DAGs constrained by MBS-learned Markov blankets to significantly outperform both SPCAND or L1MB-constrained DAGs more often than being outperformed. Using L1MB AND ($k = 4, 6$ for G50-ALL and G75-ALL, respectively) MBS outperformed SPCAND or L1MB with a greater margin. This indicates that enforcing consistency, during structure learning produces higher-quality Markov blankets, than as a post-processing step.

Table 4.3: Comparison of MBS pruning against Sparse candidate and L1 MB regularization. Legend same as Table 4.2.

		G50			G75		
		MBS	SPCAND	MBOR	MBS	SPCAND	MBOR
E	SPN	0.504	0.48	**0.458**	0.435	0.349	0.404
	1N	0.561	0.516	0.523	0.567	**0.467**	**0.523**
	2N	0.538	0.466	**0.485**	0.486	0.424	**0.474**
	3C	0.556	0.465	0.414	0.612	0.498	**0.481**
	4C	0.532	0.508	**0.463**	0.595	0.458	**0.447**
V	SPN	0.348	**0.27**	**0.27**	0.257	0.269	**0.299**
	1N	0.413	**0.295**	**0.328**	0.256	**0.288**	**0.331**
	2N	0.367	**0.274**	**0.296**	0.247	0.27	0.287
	3C	0.318	0.274	0.316	0.241	0.247	0.241
	4C	0.327	0.256	0.276	0.279	0.274	**0.22**

4.2 Scalable learning of large networks

Algorithms for inference of genome-scale networks comprising thousands of nodes are expensive in time and memory. Structure inference of biological networks is further complicated by the lack of sufficient data to reliably learn graphs over thousands of nodes. Some approaches for addressing the data sparseness problem identify variable clusters and replace the clusters by pseudo variables [125, 18]. These approaches pool the data from several variables to robustly estimate parameters. However, they do not capture the fine-grained structure within a cluster.

We present a novel, tractable approach to learn the structure of functional networks represented as undirected graphical models. Our approach, Cluster and Infer Networks (CIN), clusters the nodes into smaller groups and learns separate networks per cluster. By partitioning the nodes into smaller groups, we avoid searching over the complete node set, resulting in runtime benefits. Furthermore, learning smaller networks using the same amount of data alleviates the data sparseness problem. Because the initial clustering may not be perfect, we iteratively reassign nodes to clusters to improve the quality of the node neighborhoods, repeating the procedure until convergence. As this revisiting has extra computational cost, we make it optional by specifying the number of nodes to be revisited as an input parameter to CIN. The complete network structure is obtained by combining the networks inferred per cluster.

CIN is a meta-algorithm and can work with any existing algorithm for learning network structure. We used the Markov blanket search (MBS) algorithm for learning the structure of undirected graphs (specifically, Markov random fields). We compared CIN with and without cluster re-assignment, against standard MBS that infers networks over the entire, unpartitioned node set. This comparison was done on simulated data generated from networks of known structure. CIN gave significant speed improvements without significant accuracy loss of the inferred structures. Adding cluster reassignment further improved performance, still keeping CIN faster than MBS with no clustering.

We applied CIN to two yeast microarray compendia of glucose-starvation induced stationary phase [5]. We also compared the properties of the learned graphs against random graphs with same degree distributions as the learned graphs. Our inferred graphs had significantly more subgraphs that were enriched in at least one GO term. We found that subgraphs that were topologically close (measured by the number of edges across the subgraphs) also exhibited similar process enrichments. The correlation between topological and annotation similarity was significantly higher than random suggesting that the inferred subgraphs were representing biologically meaningful dependencies.

Overall CIN has the following advantages: (a) we provide a scalable approach to learn generic statistical dependencies in biological networks, (b) our approach captures fine-grained dependencies, which cannot be captured by simple clustering approaches, and, (c) subgraphs in our inferred networks represent significantly more biologically meaningful dependencies than in random networks.

4.2.1 Speeding up structure search using Cluster and Infer Networks (CIN)

We demonstrate the CIN framework using the Markov blanket search algorithm (MBS) described is Section 4.1.2. Due to the exponential complexity of optimal structure search, most structure learning algorithms, including MBS, learn networks with bounded neighborhood size, k [1, 40]. Although this reduces the structure search space, there are still exponentially many candidate networks to be evaluated ($O(2^{kn\log n})$) [42]. Hence, when n becomes very large (several hundreds) and k is of any interesting size ($2 \leq k \leq 10$), as is the case for genome-scale networks, these algorithms become prohibitively expensive in time and memory.

We introduce the CIN approach to further speed up structure learning of graphical models, enabling efficient learning of these models for genome-scale data. CIN comprises two steps: (a) cluster variables, and (b) infer graph structures per cluster. An optional revisit step is included where cluster assignment of variables with poor neighborhoods is updated. In CIN, variables can be clustered using any clustering approach. We use k-means with absolute value of Pearson's correlation as the similarity measure between two variables [62].

CIN is a meta-approach that works with any structure learning algorithm that identifies the best neighbor set (or parent set for Bayesian nets). We illustrate CIN with the MBS algorithm (Algorithm 2). The algorithm takes as input the gene expression data matrix $\mathcal{D} = \{\mathbf{x}^1, \cdots, \mathbf{x}^{|\mathcal{D}|}\}$, where $\mathbf{x}^d$, $1 \leq d \leq |\mathcal{D}|$, represents a joint assignment to $\mathbf{X}$ from the d^{th} datapoint. The number of clusters, c, maximum neighborhood size, k, and the number of variable neighborhoods to be revisited, r, are additional inputs to the algorithm.

The algorithm begins with partitioning the n RVs into c clusters, $\mathbf{C}_i$, $1 \leq i \leq c$. This is followed by inferring the structure per cluster $\mathbf{C}_i$ independently. In MBS this entails identifying the MB for each X_i, $\mathbf{M}_{X_i}$, such that $|\mathbf{M}_{X_i}| \leq k$ and the conditional entropy, $H(X_i|\mathbf{X}_i)$ is minimized.

The revisit step is executed if $r > 0$. This selects r variables with the worst neighborhoods (high conditional entropy in MBS) and stores them in $\mathbf{Y}$. For each $Y \in \mathbf{Y}$, we compute the change in conditional entropy, $S_{YZ} = H(Y|\mathbf{M}_Y) - H(Y|\mathbf{M}_Y \cup \{Z\})$, on adding new variables Z to $\mathbf{M}_Y$. Z is selected from clusters other than the current cluster of Y, curr_cluster(Y). $\mathbf{M}_Y$ is updated to include the variable with the maximal score improvement, Z^*. After considering all $Y \in \mathbf{Y}$, the cluster assignments are updated to incorporate the MB modifications. This in effect creates modified, possibly overlapping clusters. In the actual implementation of CIN with MBS, the revisit step is executed for every l, $1 \leq l \leq k$. We do not include this iteration in Algorithm 2 for ease of exposition.

As structures are inferred per cluster, which are much smaller than the total number of variables, we have more data to learn smaller graphs, thus producing robust structures. The speed up in CIN increases with n. Assuming that we have c clusters each of size m such that $cm = n$, the speed up is $O(\frac{m}{n}2^{kn\mathrm{log}n^a b})$, where $a = 1 - \frac{1}{c}$ and $b = c^{\frac{1}{c}}$. Thus, CIN becomes increasingly advantageous with increasing values of n.

4.2.2 Results

The goal of our experiments was two fold: (a) compare the performance of Cluster and Infer Networks (CIN) against standard graph structure learning algorithms, using both running time and quality of inferred structures, (b) evaluate the biological significance of structures inferred using CIN on real microarray data.

We address (a) using simulated data from artificial regulatory networks generated from a network simulator [115], allowing us to compare the inferred structures against ground truth. These datasets are described in Section 4.1.4. We used small networks (< 200 nodes) so that the structure could be learned using the standard graph structure learning algorithms in reasonable time. We address (b) on two microarray datasets from yeast under glucose starvation conditions [5].

CIN has significant speed benefits without substantial performance loss on simulated data

We compared three approaches to infer networks: (a) CIN with MBS with no cluster reassignment (Norevisit), (b) CIN with MBS with cluster reassignment (Revisit), and (c) standard MBS (Nocluster). For (a) and (b) we specified the number of clusters to the k-means algorithm to be 5.

We evaluated the inferred network quality using: (a) edge-wise scores to match edges

Algorithm 2 MBS with CIN

```
Input:
    Expression data of $n$ genes, $\mathcal{D} = \{\mathbf{x}^1, \cdots, \mathbf{x}^{|\mathcal{D}|}\}$
    maximum neighborhood size, $k$
    number of clusters, $c$
    convergence threshold, $\epsilon$
    number of variables for revisit, $r$
Output:
    Inferred graph structure $\mathcal{G}$.
Partition $n$ variables into $c$ clusters $\mathcal{C} = \{\mathbf{C}_1, \cdots, \mathbf{C}_c\}$, using datapoints as attributes.
while Score change $\geq \epsilon$ do
  /*Infer structure per cluster*/
  for $\mathbf{C}_i \in \mathcal{C}$ do
    Learn best structure for $\mathbf{C}_i$ s.t. $\forall X_j \in \mathbf{C}_i$, $|\mathbf{M}_{X_j}| \leq k$.
  end for
  /*Revisit*/
  if $r > 0$ then
    $\mathbf{Y} = r$ variables with worst MBs
    for $Y \in \mathbf{Y}$ do
      $S_{YZ}$=$H(Y|\mathbf{M}_Y) - H(Y|\mathbf{M}_Y \cup \{Z\})$, s.t. curr_cluster($Z$)$\neq$curr_cluster($Y$)
      $Z^* = \arg\max_{Z} S_{YZ}$
      $\mathbf{M}_Y = \mathbf{M}_Y \cup \{Z^*\}$
    end for
    Update $\mathcal{C}$ to account for MB modifications of $Y \in \mathbf{Y}$
  end if
end while
```

in the true and inferred networks, (b) pathwise scores to match edges in the true network to paths in the inferred network and vice-versa, (c) subgraph scores 1-n and 2-n subgraphs per vertex from the true network with the inferred network.

The running time of CIN with MBS, Norevisit and Revisit, is significantly smaller than standard MBS (Fig. 4.2). Further, the speed up for ECOLI (with 188 nodes) is much greater than G50 (with 100 nodes) corroborating our observation that CIN becomes increasingly advantageous with the number of nodes in the network.

The quality of the inferred networks using both CIN approaches are comparable to MBS on the complete variable set (Fig. 4.3). We show all results other than 1-n subgraphs, which is similar to 2-n. Overall, we found that CIN had significant speed benefits over standard MBS. Revisiting clusters improved results at additional runtime cost, but was still faster than standard MBS.

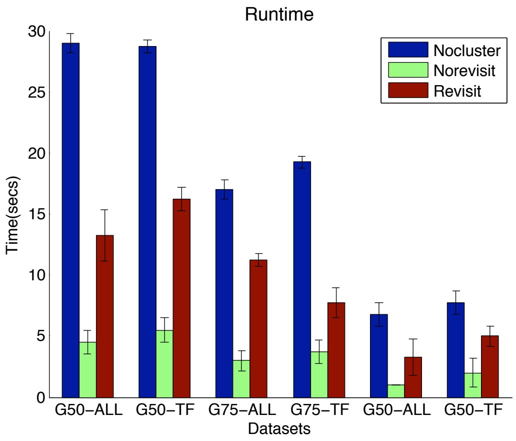


Figure 4.2: Run time for different algorithms; lower runtimes are better. Algorithms were compared using three networks of known structure. Per network, two datasets were generated by either perturbing all genes (ECOLI-ALL, G75-ALL, G50-ALL) or only transcription factor genes (ECOLI-TF, G75-TF, G50-TF).

CIN-inferred networks from microarray data have non-random topological properties

We applied CIN with MBS to two recently generated microarray compendia of yeast in stationary phase [5]. For both datasets, we specified $k = 4$ as the maximum neighborhood

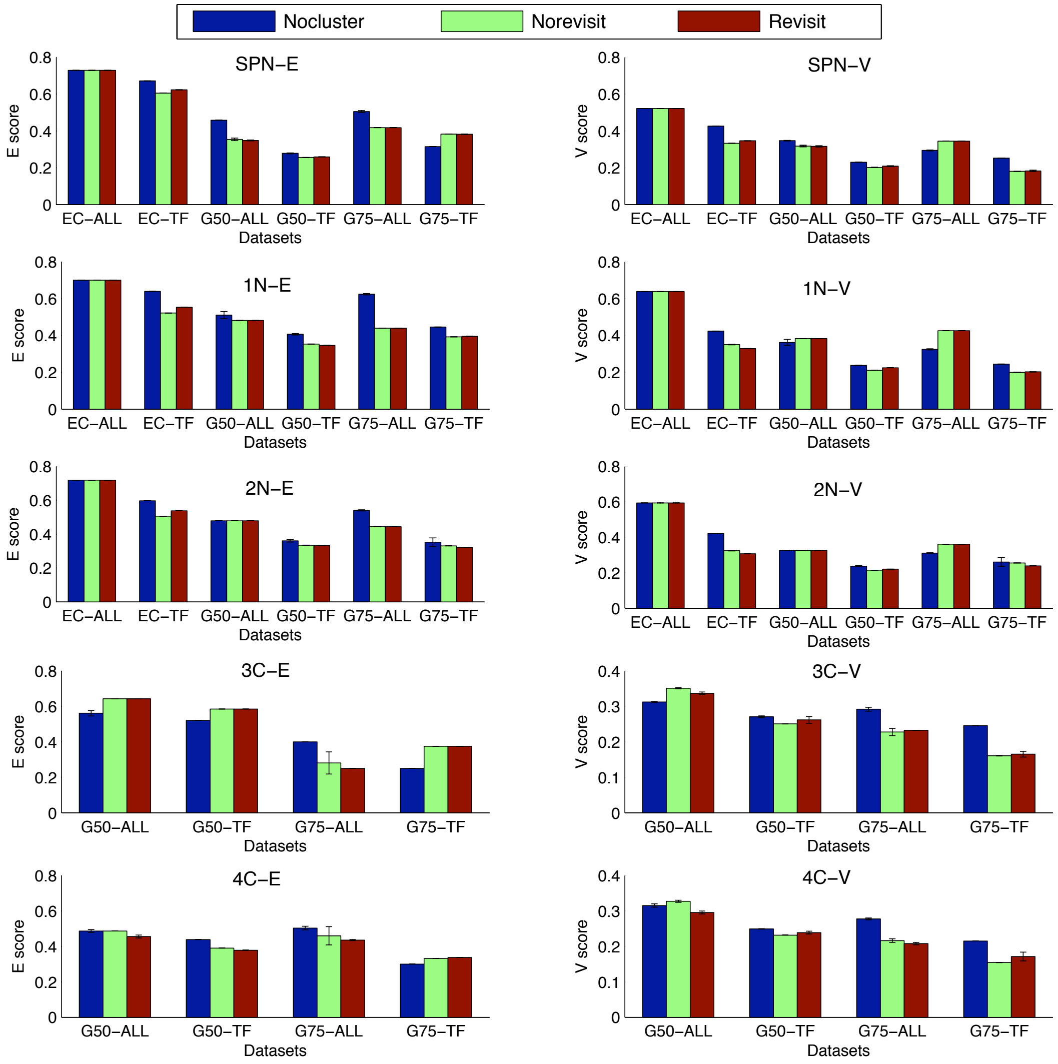


Figure 4.3: Match scores for different algorithms; higher scores are better. Nocluster: MBS algorithm without pre-clustering. Norevisit: CIN with MBS without cluster reassignment. Revisit: CIN with MBS with cluster reassignment.

size, and $c = 20$ as the number of k-means clusters. Various statistics of the clusters are reported in Table 4.4. As the true network is not known for these data, we used Gene Ontology (GO) for validation [6].

We first computed the number of 1-n subgraphs that were enriched in a Gene Ontology slim (GO slim) process term as a function of decreasing p-value (Fig 4.4). GO slim process

Dataset	Mean	Median	Stdev	Max	Min
Quiescent	140.9	146.0	46.69	214	26
Non-quiescent	140.9	147.0	77.41	300	22

Table 4.4: Different statistics of the number of genes per cluster in the quiescent and non-quiescent populations.

is a collapsed single level view of the complete GO biological process, providing high level information of the types of biological processes involving a set of genes. For both datasets, the number of subgraphs enriched in a term was significantly higher than random networks. The results belong to CIN with the revisit step. Results without the revisit step are similar.

We then computed the Annotation-Topological Similarity (ATS) at each p-value using the subgraphs that had an enrichment at that p-value (Fig 4.5). ATS provides a global measure of biological validity of the inferred networks by evaluating to see if subgraphs that are close together (small average shortest path length) also participate in the same set of biological processes. The lower the ATS score is, the more meaningful the dependencies captured in the graph.

We found that the inferred networks had lower ATS than random networks, and this was significantly lower ($p < 0.05$, simulation) at GO slim enrichment p-value, $10^{-3} < p < 0.05$. As the p-value decreased the error bars increased due to too few random subgraphs satisfying the threshold. The low ATS in the inferred networks indicates that subgraphs that are topological close, also participate in similar biological processes suggesting that the inferred topology correctly captures the functional relationships among sets of genes. Overall, the networks learned by CIN capture significantly more biologically meaningful dependencies than expected in random networks, and the global network structure of the networks captures the biological intuition that topologically close subgraphs participate in similar processes.

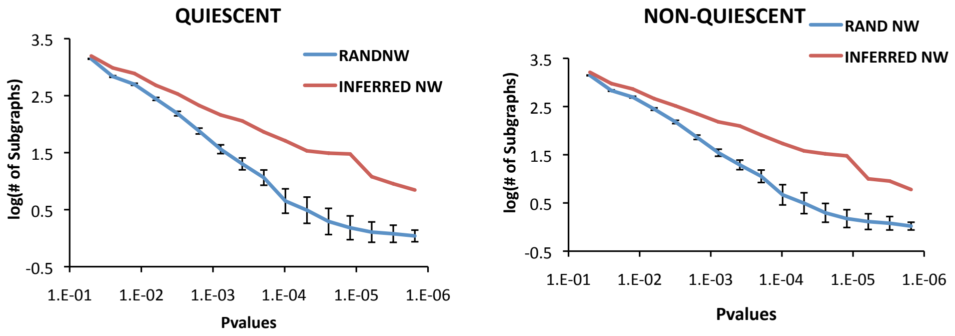


Figure 4.4: Number of subgraphs that were enriched in a GO slim process term at a specific p-value. Line with errorbars shows the average number of subgraphs enriched in random networks at that p-value. Results are shown on log-log scale.

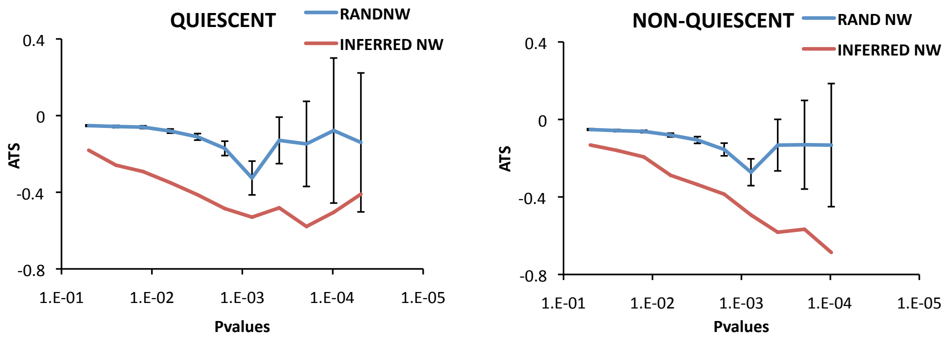


Figure 4.5: ATS measure of real and random networks at different p-values. Lower ATS is better. Results are shown on log-log scale.

4.3 Conclusion

An important first step to learning condition-specific networks is to identify the appropriate modeling framework that we will use to represent and learn condition-specfic networks. We select undirected probabilistic models because the biological networks that we want to learn are complex and are likely to contain different types of dependencies including pairwise, higher-order and cyclic dependencies. While directed graphical models such as Bayesian nets capture higher-oder dependencies, there is no easy way to capture cyclic dependencies. Undirected graphical models capture all these types of dependencies and are especially suitable when we are interested in correlative rather than causal dependencies.

We presented two novel approaches for learning undirected probabilistic graphical model structure. The first approach, Markov Blanket Search (MBS) learns the structure of MRFs by finding structurally consistent graph structures. The second approach, Cluster and Infer Networks (CIN), provides a faster approach for learning probabilistic graphical models (PGMs) for genome-scale expression data comprising thousands of nodes. At the heart of CIN is a *divide and conquer* strategy, where the problem of learning a single large graph is replaced by a set of problems, each learning smaller graphs. The MBS algorithm alone and within the CIN framework allows us to capture generic statistical dependencies among arbitrarily sized groups of genes. On simulated data with known ground truth of networks, we find that the MBS algorithm indeed is beneficial compared to the algorithms compared.

Although undirected graphical models provide a natural representation of interaction structure in complex domains such as ours, the intractability of computing the partition function poses a major challenge. We address this problem by extending an existing framework based on Markov blanket canonical parameters that allows us to efficiently learn the structure of these networks. Further, we work with approximations of likelihood, *pseudo likelihood* and information theoretic measures that allow us to avoid computing the partition function. While we do not claim that our learning algorithms produce models of as good quality if we were using data likelihood, our experiments indicate that we are able to perform equally well as existing directed models which do use likelihood based scores.

An important modeling question that arises in most probabilistic graphical models is the assumptions of the probability distribution. We performed preliminary experiments for evaluating different forms of conditional probability distributions: (a) Multi-variate Gaussians, (b) Non-parametric kernel density estimators (KDE) and (c) regression trees. Surprisingly, we found that the multi-variate Gaussians, despite their simplicity gave the best overall performance, followed closely by regression trees. Although both KDE and regression trees make fewer assumptions of the probability models than the multi-variate Gaussian, they do have hyper-parameters (kernel width in KDE and leaf node size in regression trees) that need to be tuned to model the underlying distribution correctly. We suspect that this additional complexity makes the training of these parameters difficult and may cause problems with over-fitting.

To conclude, in this chapter we have established the ground-work for learning condition-specific networks by determining that undirected graphical models provide an appropriate framework for representing biological networks. We have developed algorithms for learning the structure of these undirected models and we have also determined that conditional distributions based on multi-variate Gaussians are good choices for mathematically representing the functional dependencies between a variable and its immediate neighborhood. The work described in the subsequent chapters relies on the results of this chapter in that our algorithms use the MBS framework for representing and learning condition-specific networks and we use Gaussian-based distributions for capturing the functional dependencies among the network nodes.

Chapter 5

Higher-order dependencies: what's the deal

Statistical algorithms for learning biological networks can be roughly categorized into those that learn only pairwise dependencies: dependencies among two variables, and those that learn higher-order dependencies: dependencies among more than two variables. The ARACNE algorithm, which learns pairwise dependencies works surprisingly well on biological networks given its simplicity. Specifically, in Chapter 4 we found that ARACNE outperformed the Markov blanket search (MBS) algorithm on the ECOLI dataset. We conjectured in Chapter 4 that the ECOLI dataset does not have very many higher order dependencies and therefore the additional parameter complexity of MBS does not get balanced with the complexity of the network to be learned. This begs the question what constitutes "higher-order dependencies" in biological networks, and if there are topological characteristics of biological networks that we could associate with higher-order dependencies. While it is well-understood that protein complexes would constitute a candidate higher-order dependency, it has not been shown in practice how this affects performance of algorithms learning different types of dependencis. In this chapter, we investigate the performance of the MBS and ARACNE algorithms as a function of two topological characteristics which may represent higher-order dependencies in biological networks: (a) the in-degree of a target gene in a regulatory network, (b) the size of a transcription factor complex. Our results suggest that algorithms which learn higher-order dependencies are better for networks with nodes with high in-degree or large transcription factor complexes.

5.1 Higher-order dependencies in biological networks

Higher-order dependencies are statistical dependencies that exist among more than two nodes. Higher-order dependencies arise in biological networks via the formation of protein complexes or via interactions among transcription factors regulating a target gene.

The in-degree of a node is defined as the number of incoming edges to a node. In our simulated networks, which comprise both regulatory and protein-protein interactions, the in-degree is associated with a node corresponding to a gene and specifies the number the transcription factors regulating the gene. When a gene is regulated by more than one transcription factor, combinatorial interactions among the transcription factors may result in higher-order dependencies. These transcription factors may further physically interact via protein interactions to exert an additional layer of complexity to the regulatory program.

The RENCO simulator we developed captures the complex combinatorial interactions among the transcription factors. Here we investigate the performance behavior of two algorithms as we vary the extent of combinatorial control on the target genes. This is done using three RENCO generated datasets, in two of which we simply vary the in-degree, and in the third we keep the same in-degree but vary the number of interactions among the transcription factor proteins.

5.2 Experimental strategy

We considered three networks: ECOLI, YEAST and G75. The ECOLI and YEAST networks are a subset of the regulatory networks of the bacteria *E. coli* and the yeast *S. cerevisiea*, respectively. The G75 network is generated *de novo* from RENCO. We define a configuration parameter p, $0.1 \leq p \leq 0.9$, which influences the proportion of higher-order dependencies in the networks. For the ECOLI and YEAST networks p controls the proportion of nodes with in-degree > 1, i.e., nodes that have at least two transcription factors controlling it. A high in-degree node is that which has > 1 transcription factors regulating it. Higher values of p cause more nodes with high in-degrees.

For the G75 network, p controls the proportion of protein interactions among the transcription factors. Higher values of p cause more nodes more transcription factor cliques regulating a gene node. We vary $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ (actually for the ECOLI network we have p only till 0.7). The topological characteristics of the networks as a function of p are described in Fig 5.1. Note p controls the proportion of higher-order dependencies in the generated networks, and therefore automatically adjusts the number of higher-order dependencies as a function of the number of nodes.

We used the different scores of measuring higher-order dependencies over different folds of the data described in Chapter 3. We first considered the higher-order scores based on shortest path (SPN), and neighborhoods of radius $r = 1$ (1N) and $r = 2$ (2N). We measured both the edge set score (E score) and vertex set score (V score).

MBS and ARACNE were compared using average scores generated from structures learned on different settings of the regularization parameter λ for MBS, and the data processing inequality for ARACNE. For ECOLI we used four settings for these parameters, and G75 and YEAST we used eight settings. For each parameter setting MBS learned 11 different graphs with the maximum size of the Markov blanket, k ranging $1 \leq k \leq 11$. For ARACNE we varied the mutual information threshold to obtain 11 graphs.

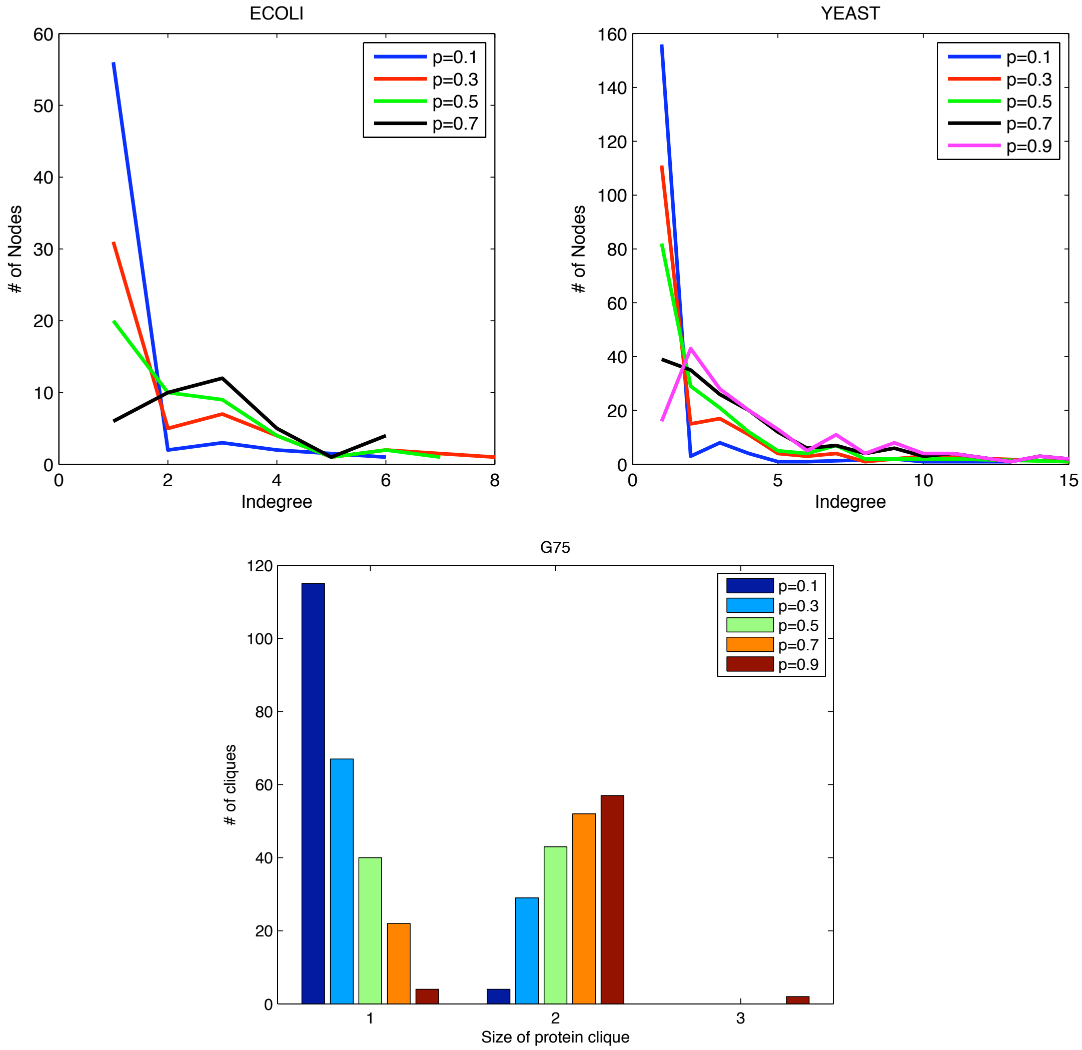


Figure 5.1: Topological properties of simulated networks as a function of p. The top two figures show the distribution of the number of nodes as a function of p for the YEAST and ECOLI networks. Higher values of p result in more nodes with high in-degree > 1. The bottom figure shows the size of transcription factor complexes for different values of p for the G75 network. Higher values of p result in more transcription factor cliques of size > 1 regulating a gene.

5.3 Results

On the ECOLI dataset (Fig 5.2), MBS has a higher performance than ARACNE, but is statistically significant for higher values of p ($p = 0.5$ for SPN E score, $p = 0.5, p = 0.7$

for 2N E score). Both algorithms had similar performance on the V score. On the YEAST dataset (Fig 5.3), both algorithms were very similar in performance, with the exception of SPN E score where MBS again was better than ARACNE for high values of p ($p = 0.5, p = 0.7$). On the G75 dataset (Fig 5.4), we found that ARACNE was significantly better on the V score, but this was concentrated on lower values of p. Specifically, for both 1N and 2N scores, we found that ARACNE was significantly better than MBS on the V score for $p = 0.3$, whereas MBS was significantly better than ARACNE for higher values of p.

We also considered the pathwise score which measures the match between an edge in the true graph with the shortest path connecting the two vertices in the inferred graph (Fig 5.5). Although there is no significant change in behavior in the ECOLI and YEAST datasets, on the G75 dataset, MBS performs significantly better when the true network has more nodes being regulated by size 2 cliques ($p = 0.7, p = 0.9$). This is consistent with our observation that high in-degree or clique size results in more higher order dependencies, which in turn is modeled better in algorithms capturing higher-order dependencies.

Overall, this suggests that as we increase the number of higher-order dependencies in the network, either by increasing the in-degree or by increasing the number of size 2 transcription factor cliques, the advantage of the MBS algorithm increases. The MBS algorithm learns higher-order dependencies and therefore as the networks get more complex in terms of the dependencies, we observe a increasing benefit of using an algorithm which learns higher-order dependencies. In contrast, when there are fewer higher-order dependencies, the ARACNE algorithm does a better job of capturing the dependencies, especially the shortest path dependencies.

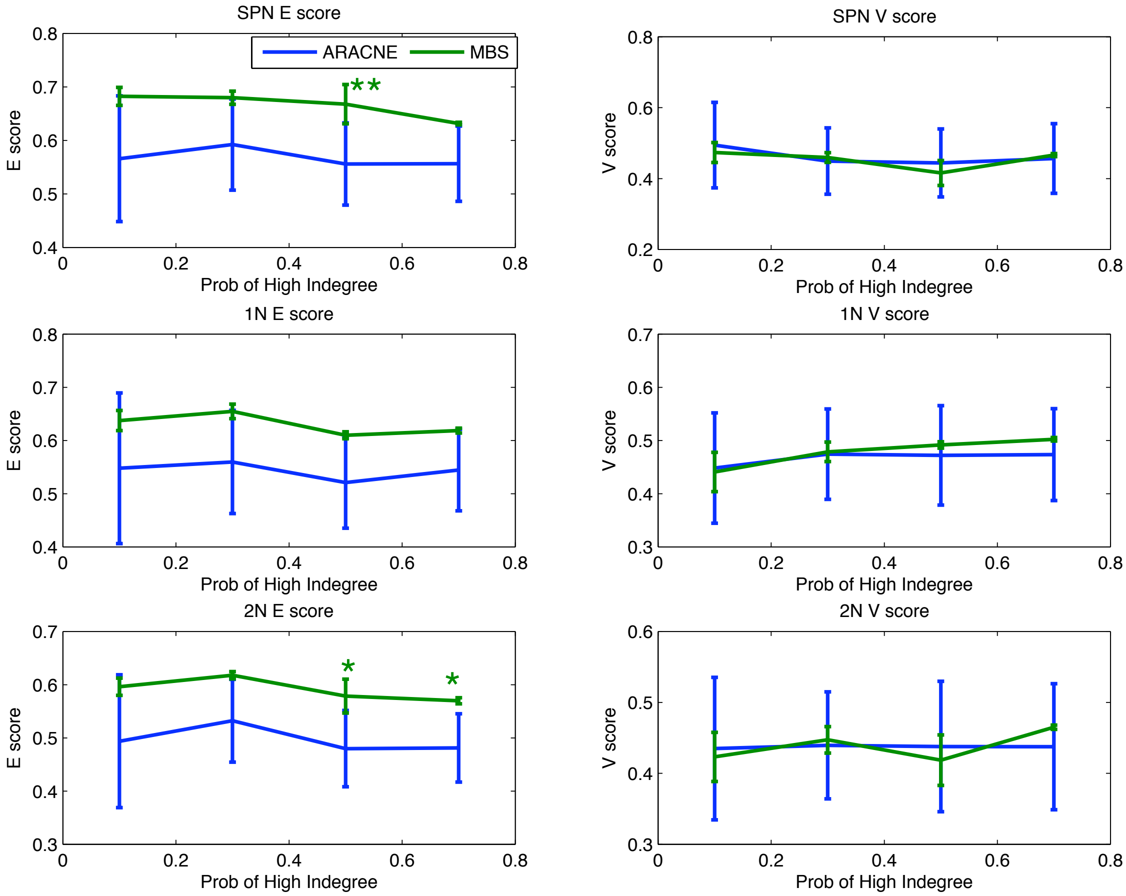


Figure 5.2: Performance of ARACNE and MBS on the ECOLI dataset. Performance using the different higher-order scores was measured as a function of increasing probability of high in-degree. * denote statistical significance, * implies $p < 0.05$ and ** implies $p < 0.01$. Green * indicates that MBS is significantly better than ARACNE and blue * indicates ARACNE is better than MBS.

5.4 Discussion

Algorithms for network inference can be roughly grouped into two categories based on the order of the dependencies they can infer. Algorithms learning pairwise dependencies are popular because of their speed benefits and also because they have fewer parameters to learn. Algorithms learning higher-order dependencies can capture more general depen-

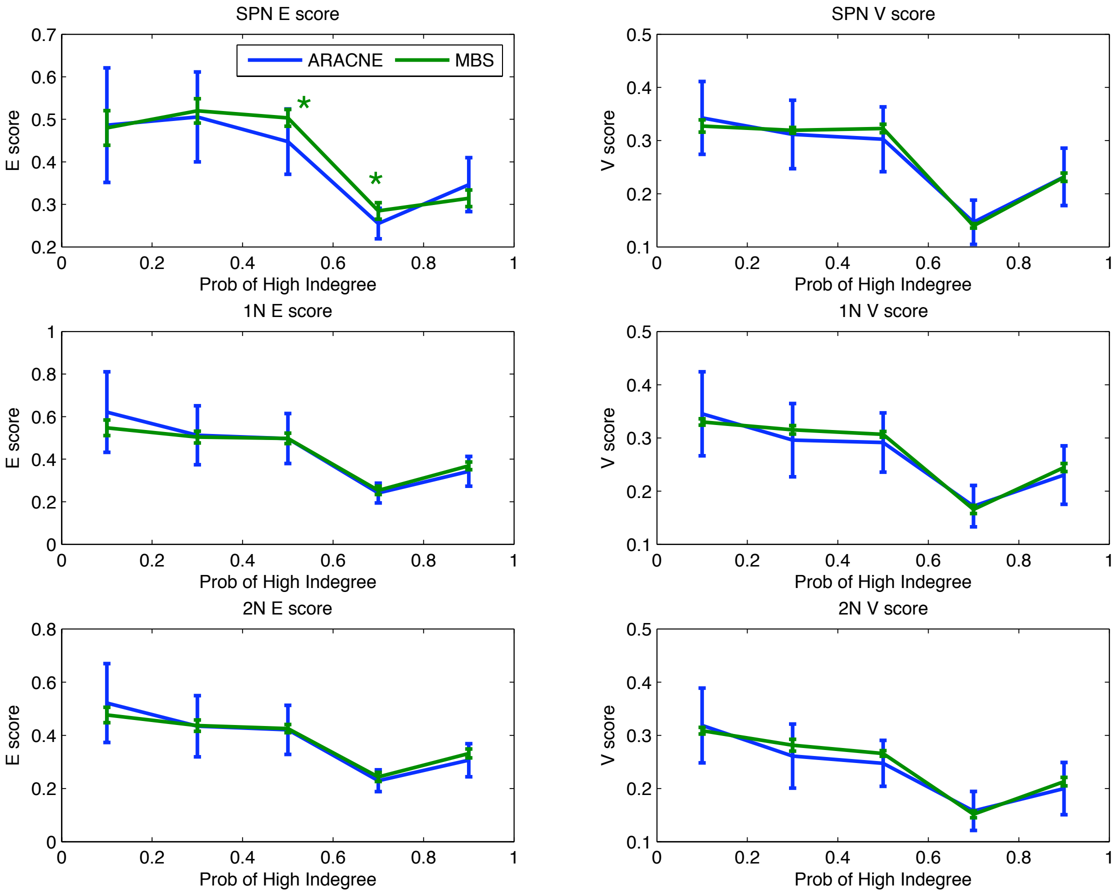


Figure 5.3: Performance of ARACNE and MBS on the YEAST dataset. Figure legend is same as in Fig 5.2.

dencies, but do so at the cost of runtime and more parameters. In this Chapter, we asked if there are certain properties of the biological network topology that are amenable to algorithms capturing either higher-order or pairwise dependencies. In particular we considered two properties: (a) the in-degree of a gene, (b) the size of TF complexes per gene.

We systematically varied the number of nodes participating in higher-order dependencies, either by increasing nodes with in-degree greater than 1, or by increasing physical interactions among the transcription factor proteins, and measured the performance of the ARACNE and the MBS algorithms. We found that it was indeed the case that as the

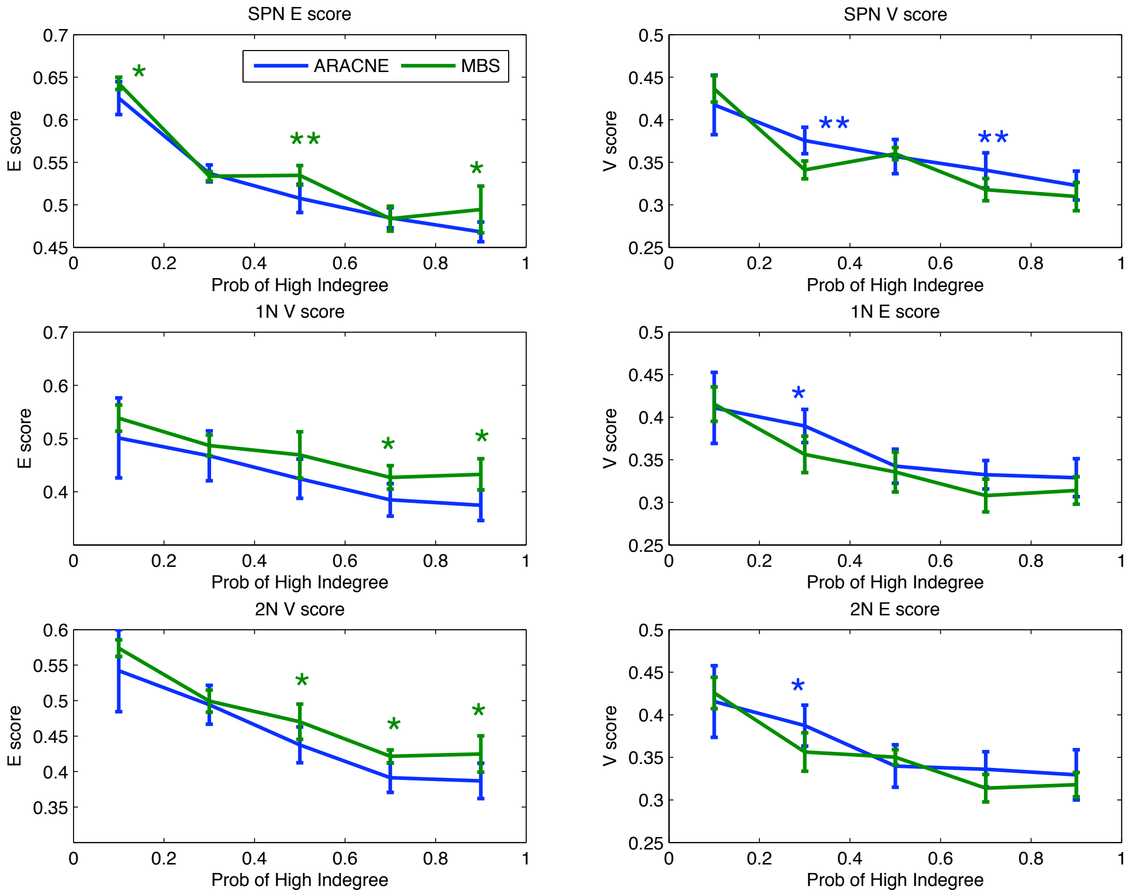


Figure 5.4: Performance of ARACNE and MBS on the G75 dataset. Figure legend is same as in Fig 5.2.

number of nodes with high in-degree nodes increased, MBS had a greater tendency to significantly beat ARACNE. A similar trend was seen also for the G75 dataset where we increased nodes being regulated by TF cliques of size > 1. In contrast, ARACNE had a tendency, albeit to a lesser extent, to beat MBS on networks with lower values of p. This suggested that MBS and other higher-order dependency learning algorithms are likely to be more beneficial when there are expected to be many nodes that are highly regulated.

Although our conjecture was validated on some cases, we were surprised that a simple pairwise learning algorithm like ARACNE performs so well. It is often assumed that

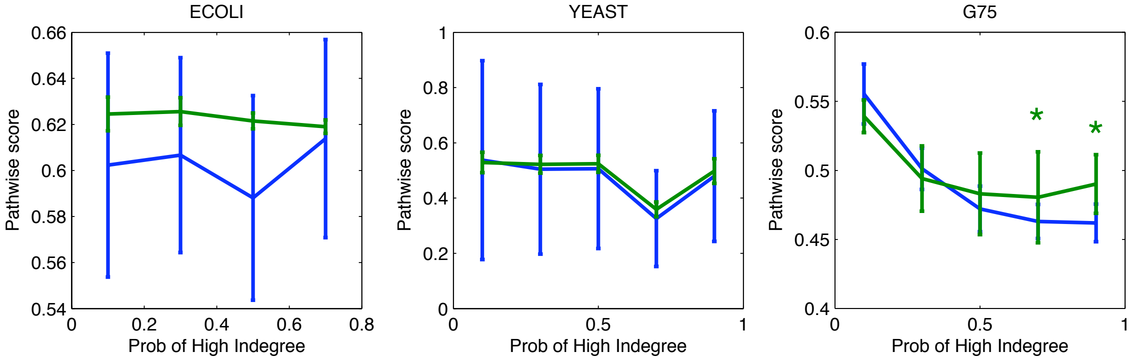


Figure 5.5: Performance comparison of ARACNE and MBS on all three datasets. Results are obtained using the pathwise score.

higher-order learning algorithms subsume and outperform pairwise algorithms. However, our experiments show that this is not universally true and there are situations where it is beneficial to use a simple pairwise algorithm. Furthermore, our experiments suggest that in-degree and the size of the transcription factor complex regulating a gene may represent higher-order dependencies, which in turn affects the performance of different algorithms.

Chapter 6

Different formulations of learning condition-specific networks

In this chapter we delve deeper into the problem of learning condition-specific networks. We first describe the simplest approach of learning networks for condition-specific response, and then present cases of increasing difficulty. In the simplest approach, the networks are learned independently from two conditions and then compared to identify similarities and differences across conditions. Next, we introduce a layer of complexity such that networks are not learned independently, but, we assume that the condition variable is observed. Finally, we describe the most general case, where conditions are not known and must be inferred during structure learning. We describe several variations on the last case and discuss the tradeoff of different models under different situations.

6.1 Learning independent networks for each condition

A straightforward approach to condition-specific networks is to learn separate networks for each condition, and then compare the network structures to identify condition-specific behavior at the network level. In this framework we can use any algorithm for learning network structures, including our Markov Blanket Search (MBS) algorithm. As a first step to condition-specific network learning, we applied MBS to learn networks from three microarray datasets in yeast. Two of these datasets measure gene expression of quiescent and non-quiescent cells under genetic mutations. The third dataset measures gene expression of exponentially growing cells to genetic and chemical perturbations. Thus, we have three conditions: two corresponding to the quiescent and non-quiescent populations respectively and the third corresponding to exponentially growing cells.

6.1.1 Data pre-processing

We applied our algorithm to two yeast, *S. cerevisiae*, datasets from quiescent and non-quiescent cells [5], and one dataset from exponentially growing cells [69]. We included only genes with $< 20\%$ missing data in all three datasets. As the quiescent and non-quiescent datasets had biological replicates, we filtered the genes further to discard not-reproducible genes. Our final datasets comprised $n = 2,818$ genes, with 170, 186, and, 300 measurements per gene in the quiescent, non-quiescent and exponential populations respectively.

6.1.2 Experimental methods

Generation of coarse modules: To obtain a high-level view of the inferred networks, we generated local subgraphs and clustered them into coarse modules. We excluded clusters of size < 5 and connected the remaining clusters into high-level graphs (Fig 6.1, Section 6.1.4).

We generated subgraphs by considering a node, and its neighbors reachable by r links. We refer to these subgraphs as 1-n subgraphs, denoting a $n$eighborhood reachable by traversing one link ($r = 1$). We computed a topological similarity for each pair of subgraphs, $\{S_i, S_j\}$, $t_{ij} = \frac{l_{ij}}{n_i+n_j}$, where l_{ij} is the sum of number of vertices common between S_i and S_j, and the number of edges across S_i and S_j. n_i and n_j are the vertex counts in subgraphs, S_i and S_j, respectively [83].

To obtain *coarse modular organization*, we first clustered the subgraphs using hierarchical clustering with average linkage [39]. We selected clusters to optimize between including majority of the genes, and to have clusters of size ≥ 5. This resulted in 230, 214 and 179 clusters, with $n = 2630, 2551$ and 2651 genes in quiescent, non-quiescent and exponential cells respectively. We then used topological similarity as edge weights for each pair of clusters.

We used the *Annotation-Topological Similarity* (ATS) measure (Chapter 3), to assess if clusters that were topologically close were also similarly annotated. Briefly, ATS is the Pearson's correlation coefficient between two vectors, v_A and v_T, each of length $\binom{|\mathbf{C}_x|}{2}$, where $\mathbf{C}_x$ is the set of clusters generated for population x. Each dimension $v_A(r)$ $(v_T(r))$ was the annotation (topological) similarity for r^{th} cluster pair, where $1 \leq r \leq \binom{|\mathbf{C}_x|}{2}$.

To compute annotation similarity, f_{ij}, for each cluster pair, $\{C_i, C_j\}$, we obtained GO slim process enrichment vector, e_i per cluster. Each dimension $e_i(r)$ was the logarithm of p-value enrichment for each process term. The annotation similarity between C_i and C_j was the Pearson's correlation coefficient between e_i and e_j.

We use the definition of topological similarity from Lee *et al.* (Chapter 3, [83]), which specifies the topological similarity for each pair $\{C_i, C_j\}$, as $t_{ij} = \frac{l_{ij}}{n_i+n_j}$, where l_{ij} is the sum of number of nodes common between C_i and C_j, and the number of edges across C_i and C_j. n_i and n_j are the vertex counts in subgraphs, C_i and C_j.

We define *relative enrichment* among two populations x and y that tests if x is equally, less or more annotated than y. Let p_y be the proportion of y's clusters that are enriched ($< 10^{-2}$) in any slim term. Assuming r out of s clusters of x are enriched, we compute the probability of observing $\leq r$ out of s using the binomial with parameter p_y. The smaller the probability the more depleted is x as compared to y. Similarly, the probability of $\geq r$ out of s enriched clusters estimates how annotated x is as compared to y. Smaller the probability the more annotated x is w.r.t to y. We repeat this for testing y's relative enrichment w.r.t x. We do this analysis for all population pairs such as quiescent versus exponential, quiescent versus non-quiescent, etc.

Identification of up or down-regulated subgraphs: We analyzed the 1-n subgraphs for *significant up-regulation* or *down-regulation* at expression level using a similar approach to Chuang *et al.* [30] Each subgraph was assigned the average of the mean expression of the subgraph genes. For each dataset and subgraph size, we estimated a null distribution of subgraph expression by randomly sampling $s = 100,000$ subsets of all genes. A p-value < 0.05 was considered as significantly up or down-regulated.

Identification of conserved and specific subgraphs: To identify *conserved subgraphs* among two cell populations, A and B, we computed a match score for each 1-n subgraph, $S_i^A \in \mathbf{S}_A$ generated from A's network, using B's network structure. This score is the harmonic mean of recall, R_i^A, and precision, P_i^A, per subgraph S_i^A. For each $S_i^A \in \mathbf{S}_A$, $R_i^A = \frac{|E_i^A \cap E_i^B|}{|E_i^A|}$, where E_i^A is the edgeset of S_i^A in A's network, and E_i^B is the edge set among S_i^A's vertices in B's network. Similarly, $P_i^A = \frac{|E_i^A \cap E_i^B|}{|E_i^B|}$. The match of S_i^A in B's network is $F_i^A = \frac{2 * P_i^A * R_i^A}{P_i^A + R_i^A}$. We assumed an $F_i^A > 0$ to indicate S_i^A is a conserved subgraph in B's network. The set of conserved subgraphs between A and B is $\mathbf{S}_A^M \cup \mathbf{S}_B^M$, where $\mathbf{S}_A^M \subseteq \mathbf{S}_A$ and $\mathbf{S}_A^M \subseteq \mathbf{S}_B$. Each $S_i^A \in \mathbf{S}_A^M$ has $F_i^A > 0$, when compared with B's network, and each $S_j^B \in \mathbf{S}_B^M$ has $F_j^B > 0$, when compared with A's network. A subgraph was considered *specific* to a particular population if it had a match score of zero for the remaining populations.

Gene ontology (GO) enrichment and false discovery rate: For each 1-n subgraph (or cluster), we used the hyper-geometric distribution to compute GO term enrichment. We sampled $s = 1000$ random subsets of size k from the $n = 2818$ genes, and computed their

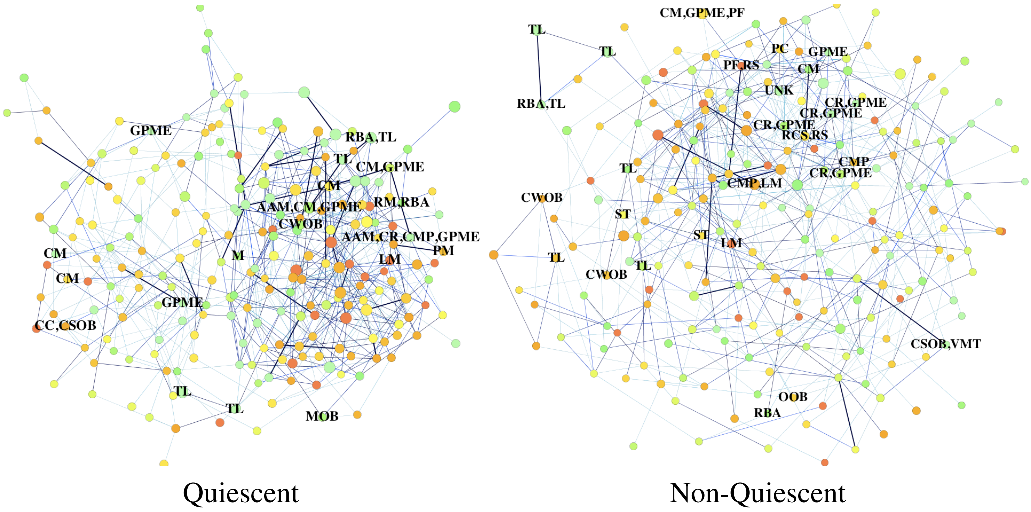
p-values for each term. The false discovery rate (FDR)[20] is associated with the number of terms enriched in a subgraph at a particular p-value. For example, if a subgraph of size k is enriched in u terms at $p < 10^{-4}$, FDR is $\frac{u'}{u}$, where u' is the average number of terms enriched in a random subgraph of size k at $p < 10^{-4}$. We used an FDR of ≤ 0.05 to identify significant enrichments.

6.1.3 Results

We applied our Markov Blanket Search (MBS) algorithm to three microarray datasets, measuring gene expression of quiescent, non-quiescent and exponential cells. We analyzed the inferred networks to identify coarse and fine-grained network properties that discriminate the quiescent from non-quiescent cells, and both of these cells from exponential cells. Specifically, we performed: (a) a high-level cluster analysis of the inferred networks to identify coarse differences, (b) fine grain analyses of the subgraphs using Gene ontology process, and (c) analysis of hubs in the inferred networks.

6.1.4 Modular organization in quiescent, non-quiescent and exponential cells.

To obtain a high-level view of inferred networks, we clustered the subgraphs from inferred networks per population, followed by GO slim process enrichment of the clusters (Figs 6.1,6.2). We found good correlation between annotation and topological similarity for exponential (0.51) and non-quiescent cells (0.49), suggesting that similarly functioning genes are topologically close in the inferred networks. Quiescent cells had relatively lower correlation (0.26), suggesting that gene expression is more informative for non-quiescent and exponential cells than for quiescent cells. Quiescent cells may employ additional mechanisms, including post-translational modification, to respond to stresses [4, 5]. The



Quiescent

Non-Quiescent

Figure 6.1: Coarse modular organization of networks inferred from the three populations. Each node represents a cluster of genes. The node size is proportional to the gene count per cluster. The color indicates if cluster genes are more expressed (red) or repressed (green). The edge thickness is proportional to the similarity between two clusters. The nodes are labeled with processes enriched in the member genes. AAM: amino acid and deriv. ACM: aromatic comp CM: carb. meta CB: cell budding CC: cell cycle CH: cellular homeostasis CR: cellular resp. CWOB: cell wall org. & biogen. CMP: cofactor metab. C: conjugation CK: cytokinesis CSOB: cytoskeleton org. & biogen. DM: DNA metab. process GPME: gen. of pre. metab. &energy HM: heterocycle metab. LM: lipid metab. M: meiosis MOB: membr. org. & and biogen. NOB: nuclear org. & biogen. OOB: organelle org & biogen. UNK &other PC: protein catab. PF: protein folding PM: protein modif PG: pseudohyphal growth RCS: resp. to chemical stimulus RS: response to stress RBA: ribosome biogen & assem RM: RNA metab ST: signal transduction SP: sporulation TR: transcription TL: translation T: transport TP: transposition VMT: vesicle-mediated transport VM: vitamin metabolic process.

presence of post-translational modification is further supported by up-regulation of a quiescent cluster enriched in protein modification (PM).

Both quiescent and non-quiescent cells have significantly fewer annotated clusters than exponential (Table 6.1). Quiescent is also less annotated than non-quiescent. This highlights the fact that yeast quiescence is a relatively under-studied process, motivating further investigation of these cells.

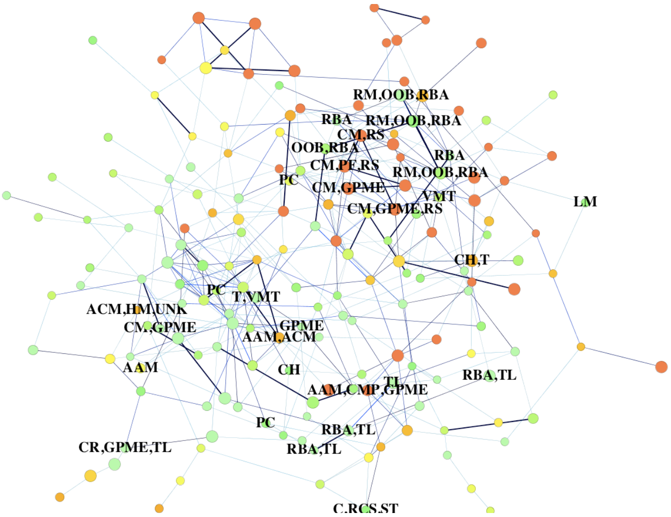


Figure 6.2: Coarse modular organization in the network inferred from exponentially growing cells. Figure legend is the same as Fig 6.2

Population	Enriched/Total clusters	wrt EXP		wrt Q		wrt NQ	
		↑	↓	↑	↓	↑	↓
Exponential	32/179	–	–	3e-4	1.0	0.02	0.98
Quiescent	19/230	1	7e-5	–	–	0.98	0.02
Non-quiescent	27/214	0.97	0.03	0.01	0.99	–	–

Table 6.1: Relative enrichment of clusters. ↑ and ↓ denote enrichment and depletion p-value, respectively, of annotated clusters. Each row corresponds to a population and specifies the relative enrichment of all other populations with respect to this population. See Section 6.1.2 for details.

6.1.5 Fine grained analysis of the cell populations

To identify similarities and differences among the populations, we obtained GO process enrichment for individual 1-r subgraphs.

Population pair	Subgraphs	Enr	Up	Down	Enr & Up	Enr & Down
Q-NQ	833	97	26	95	4	27
NQ-EXP	288	99	7	25	5	16
Q-EXP	311	98	11	27	0	24

Table 6.2: Number of common subgraphs, where two subgraphs are considered common using per-subgraph match score. Enr is the number of subgraphs enriched in a gene ontology process. Up and Down specifies the number of subgraphs that are significantly up or down regulated, respectively, in gene expression. Up & Enr are the number of subgraphs that are both enriched in a gene ontology process *and* up regulated in gene expression. Down & Enr are the number of subgraphs that are both down regulated *and* enriched in gene expression.

Processes that are similarly enriched in quiescent and non-quiescent cells indicate global starvation response

To identify similarly enriched processes, we obtained conserved subgraphs among two populations. For each subgraph, we assessed GO process enrichment, and whether it agreed in expression – both up or both down-regulated – in the two populations being compared.

There were a large number subgraphs common between quiescent and non-quiescent (Table 6.2). These subgraphs were enriched in glycolysis, fermentation, translation and fatty acid oxidation processes. However, only half agreed in expression. Several of these subgraphs had both positive and negative correlation, resulting in an overall subgraph expression that was neither significantly high nor low. These heterogeneous dependencies indicate more complex relationships not likely to be captured by co-expression.

We also identified several subgraphs conserved between quiescent and non-quiescent populations, that were up-regulated, but did not have term enrichment. The majority of the genes from these subgraphs were associated with unknown biological process, emphasizing the importance of studying these cells.

Non-quiescent and exponential cells had several common subgraphs enriched in telom-

Population	Subgraphs	Enr	Up	Down	Enr and Up	Enr and Down
Q	2295	70	174	160	7	17
NQ	2317	74	171	186	10	11
EXP	2570	232	327	206	54	44

Table 6.3: Subgraphs specific to individual populations. Same legend as Table 6.2.

Population	Process	genes
Q	RAS signal transduction	IRA1, SPG3, YGR026W, BCY1, PFK2
	Sporulation	GPA2, GSC2, OSW2, CAF120, YOR277C
	de novo pyrimidine base biosynthetic process	ARF1,DIG1, URA1,URA3,YHR003C
NQ	hyperosmotic response	TRS120, MSB2, YHR100C, PBS2, RSF2
	regulation of DNA metabolic process	PIM1, DIG2, BCK2,SSL2, DPB11, PLB3, HST1, YNG1
EXP	ATP biosynthesis	COX20, QCR10,QCR8, ATP2, ATP7
	cell wall organization & biogenesis	AFR1, SKN1,GFA1, KTR2, DFG5
	amino acid biosynthesis	IDP1, ARO3, HOM2, YGL117W, YSC83, ARG4,SIP4, CPA2,ARG1, SER1, SSU1
	response to toxin	AAD10,AAD16, AAD4,BAP2,MID2, TAT1,TYR1

Table 6.4: Processes exclusively up regulated in different populations

ere maintenance, DNA packaging, chromatin assembly and mitotic recombination. These findings are consistent with previous knowledge that non-quiescent cells have unstable genomes due to damaged DNA and can rapidly incur mutations [3]. Comparison of quiescent and exponential cells did not identify any processes enriched in the up-regulated subgraphs. The processes enriched in down-regulated subgraphs included glycolysis, gluconeogenesis and ribosomal biogenesis.

Overall, the subgraph analysis suggests that quiescent and non-quiescent cells are more similar to each other than either is to exponential cells. There are several subgraphs common to quiescent and non-quiescent cells, but not all agree in expression. The processes that are common between these cells suggest global environmental response as the cells transition from fermentable to non-fermentable carbon sources for energy.

Differences in quiescent and non-quiescent cells suggest population-specific response

We examined GO enrichment of subgraphs that occurred only in one population. Both quiescent and non-quiescent cells had fewer subgraphs with enriched processes than exponential (Table 6.3). The quiescent cells were exclusively enriched in sporulation and negative regulation of the RAS signal transduction pathway (Table 6.4). Down regulation of this pro-growth pathway indicates mechanisms to conserve energy expended in growth conditions. Furthermore, subgraph genes that are not annotated with signal transduction (SPG3, PFK2), are all important for stationary phase. SPG3 is required for survival under high temperatures and PFK2 is essential for anaerobic growth.

The non-quiescent cells exhibited processes involved in osmotic stress response and regulation of DNA recombination. This is consistent with these cells trying to cope with environmental changes and that they have unstable genomes. However, unlike quiescent cells, most of the processes up-regulated in non-quiescent cells, also occurred in exponential cells.

The exponential cells were enriched in response to chemical stresses, biosynthesis of amino acids and ATP biosynthesis. ATP biosynthesis was down-regulated in both quiescent and non-quiescent cells. The up-regulation of these energy producing pathways suggests that exponential cells expend a large amount energy to make relevant mRNA in response to different stresses. In contrast, as quiescent cells are formed in response to a starvation condition, they are likely to sequester mRNA for rapid release in response to different stresses [4].

Non-quiescent hubs are enriched in disease causing genes

We analyzed the inferred networks to identify network hubs, nodes with degree ≥ 7. A significant overlap between quiescent and non-quiescent hubs (n=29) implied similarities among these cells due to global starvation response, consistent with Section 6.1.5.

Population	Hubs	Exclusive	Processes
Q	215	167	cell wall organization & biogenesis signal transduction, carbohydrate metabolism, organelle organization and biogenesis, generation of precursor metabolites
NQ	166	116	vesicle-mediated transport, response to stress, membrane organization & biogenesis
EXP	318	273	aminoacid & derivative process, cellular respiration ribosome biogenesis & assembly

Table 6.5: Hub nodes and their most enriched processes

The non-quiescent cells have been hypothesized as models for studying diseases in humans due to the instability of their genomes [5]. We asked if hubs from different cell populations were enriched in human disease causing gene homologs [60] (Table 6.6). Of the $n = 2818$ genes used to infer networks, there were $n = 225$ yeast genes, homologous to different human disease genes[1]. We found that hubs in non-quiescent cells are more likely to be enriched in disease homologs than either quiescent or exponential cells. This provides preliminary empirical evidence for the hypothesis that these cells can provide insight into human disease causing conditions.

We found network hubs from quiescent cells to be enriched (p-value <0.05) in several GO slim processes such as signal transduction and cell wall biogenesis (Table 6.5). Among the quiescent hubs was SNF1, known to be crucial for the formation of quiescent cells. The non-quiescent hubs were enriched in stress response and vesicle mediated transport. Finally the exponential hubs were enriched in amino acid processes and cellular respiration. The enrichment of different processes further illustrates the underlying biochemical characteristics that discriminate these cells, and how they respond to different stresses.

[1] We downloaded human-yeast homologs from `http://www.biomart.org/index.html`

Population	Total Hubs	Homologous Disease Hubs	Pval
NQ	166	26	1e-4
Q	215	22	0.147
EXP	318	10	0.395

Table 6.6: Enrichment of human disease gene homologs in hubs.

6.2 Learning condition-specific networks that incorporate shared information

In this framework our goal is to exploit the shared information across conditions during network structure learning. We consider two variations of this problem: in the first, the condition variable is known and we describe algorithms that exploit the shared information by using a new kind of parameterization of the conditional probability distributions. In the second, the condition variable is hidden and we develop several models based on mixtures of graphs that automatically take into account the sharing of information across conditions.

6.2.1 Network Inference with Pooling Data (NIPD)

The NIPD framework assumes that condition variables are observed and uses a novel score that evaluates candidate networks with respect to data from any subset of conditions, pooling data for subsets with more than one conditions. To motivate the NIPD approach, we consider a hypothetical example from a biological setting. Consider two yeast environmental stimuli, with glucose (G+) and without glucose (G-). A regulatory yeast gene, HAP4 can regulate a target COX8 either only in G+ or in both G+ and G-[2]. Thus, the edge $\{\text{HAP4} - \text{COX8}\}$ occurs in the singleton condition set $\{\text{G+}\}$ as well as in set $\{\text{G+}, \text{G-}\}$. In general, the NIPD approach works on the following idea: for an edge to exist in condition, c, it can exist in any subset $\mathbf{C} \subseteq \mathcal{C}$ such that $c \in \mathbf{C}$, where $\mathcal{C}$ is the set of conditions.

[2] This is a completely synthetic example and is used here only for the sake of exposition

This approach of enumerating over all subsets allows us to simultaneously identify the edges that are present exclusively in one condition, and also the edges that are shared in any *subset* of conditions. In essence, this means we have as many networks as the size of the power set of $\mathcal{C}$. This obviously raises the question of the computational and statistical feasibility of learning so many networks. To deal with the computational complexity we restrict k to a small number. To deal with the statistical issue of sparse data, we pool the data for any non-singleton condition case and we use conditional probability distributions that capture the pooling affect. We now describe the problem more formally.

Probabilistic graphical modeling framework

Let $\mathcal{C}$ denote the set of k conditions and let $D_1, \cdots D_k$ denote the datasets for each condition. Our goal is to learn k graphs, $G_1, \cdots, G_k$ for the k conditions. The problem of learning these k graphs simultaneously can be solved in a Bayesian framework of maximizing the posterior probability of the hidden graphs, $G_1, \cdots, G_k$, $P(G_1, \cdots, G_k|D_1, \cdots, D_k)$. Applying Bayes rule:

$$P(G_1, \cdots, G_k|D_1, \cdots, D_k) \propto P(D_1, \cdots, D_k|G_1, \cdots, G_k)P(G_1, \cdots, G_k) \tag{6.1}$$

$P(D_1, \cdots, D_k|G_1, \cdots, G_k)$ is the likelihood of data given the model. $P(G_1, \cdots, G_k)$ is a prior over graph structures that biases the search of graphs. Typically a uniform prior or a prior that favors less complex structures is used. We use the minimum description length (MDL) principle to selects graphs that give good data likelihood and are the simplest in complexity. A full Bayesian solution to computing $P(D_1, \cdots, D_k|G_1, \cdots, G_k)$ will require us to solve the integral:

$$\begin{split} P(D_1, \cdots, D_k|G_1, \cdots, G_k) = \int P(D_1 \cdots, D_k|\theta_1, \cdots, \theta_k) \\ P(\theta_1, \cdots, \theta_k|G_1, \cdots, G_k)d\theta_1, \cdots, \theta_k \end{split} \tag{6.2}$$

where θ_c, $1 \leq c \leq k$ is the parameter set for the l^{th} graph. Instead we use a most likely point estimate of the parameters, aka maximum likelihood (ML) estimates, allowing us to compute the likelihood as:

$$P(D_1,\cdots,D_k|G_1,\cdots,G_k) = P(D_1,\cdots,D_k|\widehat{\theta}_1,\cdots,\widehat{\theta}_k)P(\widehat{\theta}_1,\cdots,\widehat{\theta}_k|G_1,\cdots,G_k),$$

where $\widehat{\theta}_1,\cdots,\widehat{\theta}_k = \underset{\theta_1,\cdots,\theta_k}{\arg\max}\, P(D_1\cdots,D_k|\theta_1,\cdots,\theta_k)P(\theta_1,\cdots,\theta_k|G_1,\cdots,G_k)$

To avoid clutter in notation, we drop the distinction between θ_c and $\widehat{\theta}_c$ and let θ_c denote the ML estimates. This gives the following score for the k graphs in the MDL framework:

$$S(G_1,\cdots,G_k) = P(D_1,\cdots,D_k|\theta_1,\cdots,\theta_k)P(\theta_1,\cdots,\theta_k|G_1,\cdots,G_k) - \text{MDL Penalty} \tag{6.3}$$

We assume $P(D_c|\theta_1,\cdots,\theta_k) = P(D_c|\theta_c)$, that is, D_i are independent of each other, given their respective parameters. Thus, $P(D_1,\cdots,D_k|\theta_1,\cdots,\theta_k) = \prod_{c=1}^{k} P(D_c|\theta_c)$. Because our graphs are undirected graphs, we use pseudo likelihood instead of data likelihood [15].

We expand the complete condition-specific parameter set θ_c, to $\{\theta_{c1},\cdots,\theta_{cN}\}$. Each θ_{ci} specifies the parameters of the conditional distribution of each variable X_i, $1 \leq i \leq N$ and its neighborhood (Markov blanket) in condition c. We make the typical parameter independence assumption for each variable [64], which states that the parameters of each conditional distribution are independent of each other given the Markov blanket of each variable:

$$P(\theta_1,\cdots,\theta_k|G_1,\cdots,G_k) = \prod_{i=1}^{N} P(\theta_{1i},\cdots,\theta_{ki}|\mathbf{M}_{1i},\cdots,\mathbf{M}_{ki}) \tag{6.4}$$

Here $\mathbf{M}_{ki}$ denotes the Markov blanket of variable X_i in condition c and θ_{ci} represents the parameters of the conditional distribution of $P(X_i|\mathbf{M}_{ci})$. Note the parameters of conditional probabilities of individual random variables are independent, but we have not made any independence assertions about the parameters per variable across conditions. Ideally, we need a prior distribution over the parameters such that the dependence between the parameters per condition are obtained. We do not have a way to specify this prior probability distribution over the θ_{ci}, but instead we specify the structure of θ_{ci} in a way that enforces dependencies among the parameters across conditions.

To enforce dependency among the θ_{ci}, we make $\mathbf{M}_{ci}$ depend on all the neighbors of X_i in condition c and all sets of conditions that include c. To convey the intuition behind this idea, let us consider the two condition case $\mathcal{C} = \{A, B\}$. A variable X_j can be in X_i's MB in condition A, either if it is connected to X_i only in condition A, or if it is connected to X_i in both conditions A and B. Let $\mathbf{M}^*_{Ai}$ be the set of variables that are connected to X_i only in condition A but not in both A and B. Similarly, let $\mathbf{M}^*_{\{A,B\}i}$ denote the set of variables that are connected to X_i in both A and B conditions. Hence, $\mathbf{M}_{Ai} = \mathbf{M}^*_{Ai} \cup \mathbf{M}^*_{\{A,B\}i}$. For each of these MBs, we have parameters, θ^*_{Ai} and $\theta^*_{\{A,B\}i}$, for the conditional distributions $P(X_i|\mathbf{M}^*_{Ai}, \theta^*_{Ai})$ and $P(X_i|\mathbf{M}^*_{\{A,B\}i}, \theta^*_{\{A,B\}i})$. Because θ_{Ai} depends upon $\mathbf{M}_{ci}$, $\theta_{Ai} = \{\theta^*_{Ai}, \theta^*_{\{A,B\}i}\}$. Similarly, for condition B, $\mathbf{M}_{Bi} = \mathbf{M}^*_{Ai} \cup \mathbf{M}^*_{\{A,B\}i}$, implying $\theta_{Bi} = \{\theta^*_{Bi}, \theta^*_{\{A,B\}i}\}$. The dependency between the parameters of X_i in A and B, θ_{Ai} and θ_{Bi}, is imposed by the shared dimension $\theta^*_{\{A,B\}i}$.

More generally, for any $c \in \mathcal{C}$, $\mathbf{M}_{ci} = \bigcup_{\mathbf{E} \in \text{powerset}(\mathcal{C})\,:\,c \in \mathbf{E}} \mathbf{M}^*_{\mathbf{E}i}$, and $\theta_{ci} = \bigcup_{\mathbf{E}\,:\,c \in \mathbf{E}} \theta^*_E$. Here $\mathbf{M}^*_{\mathbf{E}i}$ denotes the neighbors of X_i *only* in condition set $\mathbf{E}$. To incorporate this dependency in the structure score, we need to define $P(X_i|\mathbf{M}_{ci})$ such that it takes into account all subsets $\mathbf{E}$, $c \in \mathbf{E}$. We propose two forms of representing the conditional probability distribution: product model and weighted sum model.

The product model defines the conditional probability distribution of a variable X_i given its Markov blanket $\mathbf{M}_{ci}$ as:

$$P(X_i|\mathbf{M}_{ci}, \theta_{ci}) \propto \prod_{\mathbf{E} \in \text{powerset}(\mathcal{C}) : c \in \mathbf{E}} P(X_i|\mathbf{M}^*_{\mathbf{E}i}, \theta^*_{ci}), \tag{6.5}$$

This model assumes that the MBs, $\mathbf{M}^*_{\mathbf{E}i}$, independently influence X_i. This allows us to write $P(X_i|\mathbf{M}_{ci})$ as a product: $P(X_i|\mathbf{M}_{ci}) \propto \prod_{\mathbf{E} \in \text{powerset}(\mathcal{C})\,:\,c \in \mathbf{E}} P(X_i|\mathbf{M}^*_{\mathbf{E}i})$. We refer to NIPD with product model as NIPD-PROD. The proportionality sign can be eliminated by dividing the product by a normalization constant. Assuming a conditional Gaussian for the form of the conditional probability distribution, this normalization constant can be obtained by writing out the complete RHS of Eq 6.5, which has the form:

$$\mathcal{N}\left(x_{1id}|\frac{\mu_{1id}\sigma^2_{3i}+\mu_{3id}\sigma^2_{1i}}{\sigma^2_{3i}+\sigma^2_{1i}};\frac{\sigma_{1i}\sigma_{3i}}{\sqrt{\sigma^2_{1i}+\sigma^2_{3i}}}\right)\mathcal{N}(\mu_{1id}|\mu_{3id};\sqrt{\sigma^2_{1i}+\sigma^2_{3i}})$$

The first Gaussian gives the normalized conditional probability distribution and the normalization term is $\frac{1}{Z_{1id}}=\mathcal{N}(\mu_{1id}|\mu_{3id},\sigma^2_{1i}+\sigma^2_{3i}))$, where σ^2_{3i} is the standard deviation from the condition set $\{1,2\}$, $\mu_{1id}=\mathbf{w}^{\mathsf{T}}_{1i}\mathbf{m}^{*}_{1id}$, is the mean of the conditional Gaussian using the d^{th} data point in condition 1 (See Appendix B).

However, we work directly with the product of conditionals, that is, the un-normalized conditional probability distribution for two reasons: (a) the score improvement can be computed very efficiently, and (b) the second Gaussian acts as a smoothing term over the parameter μ_{1id}. In particular, if we were estimating a new parameter μ_{1id}, the second Gaussian specifies the probability of the new μ_{1id} using a Gaussian centered around the mean computed from the pooled dataset preferring network structures with means μ_{1id} closer to the shared mean μ_{3id}. Our preliminary experiments showed that this score has better performance than if we were to subtract out the normalization term.

The weighted sum model uses a linear combination model that sums up the conditional probability of X_i from all the relevant condition sets:

$$P(X_i|\mathbf{M}_{ci})=\sum_{\mathbf{E}\in\text{powerset}(\mathcal{C}):c\in\mathbf{E}} w_{\mathbf{E}}P(X_i|\mathbf{M}^{*}_{\mathbf{E}i}), \tag{6.6}$$

where $\sum_{\mathbf{E}:c\in\mathbf{E}}w_{\mathbf{E}}=1$, and $w_{\mathbf{E}}>0$. The above expression can be interpreted as an expected conditional probability, where the $w_{\mathbf{E}}$ represent a (prior) probability over the different MBs, $P(M^{*}_{\mathbf{E}i}):P(X_i|\mathbf{M}_{ci})=\sum_{\mathbf{E}}w_{\mathbf{E}}P(X_i|\mathbf{M}^{*}_{\mathbf{E}i})=E_{P(\mathbf{M}^{*}_{\mathbf{E}i})}[P(X_i|M^{*}_{\mathbf{E}i})]$.

The weights $w_{\mathbf{E}}$ can be set a variety of ways. We consider two scenarios: (a) fixed weights (NIPD-WTSUM), (b) learned weights (NIPD-WT-LEARN). For (a) we set the weights to be equal. For (b), weights are estimated for each variable and are proportional to the contribution of the variable to the overall pseudo likelihood of the data from the condition set:

$$w_{\mathbf{E}} \propto \sum_{d=1}^{|D_{\mathbf{E}}|} P(X_i = x_{id}|\mathbf{M}_i = \mathbf{m}_{id}), \text{where } D_{\mathbf{E}} = \cup_{e\in\mathbf{E}} D_e. \tag{6.7}$$

We refer to NIPD with fixed and learned weights as NIPD-WTSUM and NIPD-WT-LEARN, respectively.

This weighted sum definition of the conditional probability distribution is motivated by a voter model where we have several predictors for the test variable (X_i in our case) [27], and we take an average of the predicted values of X_i from all the individual predictors. This model is also similar to a hierarchical mixture of experts model [70], where data from each class is generated via a mixture model, but the components of the mixture are shared across classes.

During structure learning (described in detail in the next section) we maintain a conditional distribution for every RV, X_i, for every set $\mathbf{C} \in$ powerset($\mathcal{C}$). We consider the addition of an edge $\{X_i, X_k\}$ in every set $\mathbf{C}$ and compute the change in the score due to addition of an edge. This edge addition will affect the conditionals of X_i and X_j in all conditions $c \in \mathbf{C}$. Let PLLV denote X_i's contribution to the overall pseudo-likelihood and is defined, including a minimum description length (MDL) penalty, as PLLV$(X_i, \mathbf{M}_{ci}, c) = \sum_d^{|D_c|} \log P(X_i|\mathbf{M}_{ci}) + \frac{|\theta_{ci}|\log(|\mathcal{D}_c|)}{2}$. The net score improvement of adding an edge $\{X_i, X_j\}$ to a condition set $\mathbf{E}$ is given by:

$$\begin{aligned}\Delta\text{Score}_{\{X_i,X_j\},\mathbf{C}} = \sum_{e\in\mathbf{C}} \sum_{d=1}^{|\mathcal{D}_e|} &\text{PLLV}(X_i, \mathbf{M}_{ei} \cup \{X_j\}, e) - \text{PLLV}(X_i, \mathbf{M}_{ei}, e) + \\ &\text{PLLV}(X_j, \mathbf{M}_{ej} \cup \{X_i\}, e) - \text{PLLV}(X_j, \mathbf{M}_{ej}, e)\end{aligned} \tag{6.8}$$

Depending upon which definition of $P(X_i|\mathbf{M}_{ci})$ we use, we will have different ways of computing the score improvement. We first consider the product model. Note in this case Markov blanket variables per condition set independently influence the conditional,the pseudo-likelihood $\text{PLLV}(X_i, \mathbf{M}_{ei}, e)$ decomposes as $\sum_{\mathbf{E}\text{ s.t: } e\in\mathbf{E}} \text{PLLV}(X_i, \mathbf{M}^*_{\mathbf{E}i}, e)$. Because of this decomposability, all terms other than those involving the Markov blanket variables in condition set $\mathbf{E}$ remain unchanged producing the score improvement:

$$\begin{aligned}\Delta\text{Score}_{\{X_i,X_j\},\mathbf{C}} = {} & \text{PLLV}(X_i, \mathbf{M}^*_{\mathbf{C}i} \cup X_j, \mathbf{C}) - \text{PLLV}(X_i, \mathbf{M}^*_{\mathbf{C}i}, \mathbf{C}) \\ & + \text{PLLV}(X_j, \mathbf{M}^*_{\mathbf{C}j} \cup X_i, \mathbf{C}) - \text{PLLV}(X_j, \mathbf{M}^*_{\mathbf{C}j}, \mathbf{C}) \qquad (6.9)\end{aligned}$$

This score decomposability allows us to efficiently learn networks over condition sets.

Now we consider the weighted sum model. In this case the pseudo likelihood has a log of sums and does not decompose over condition sets. Specifically the pseudo likelihood contribution of each RV, X_i is given by

$$\text{PLLV}(X_i, \mathbf{M}_{ci}, c) = \sum_{d}^{|D_c|} \log\left(\sum_{\mathbf{E}\in\text{powerset}(\mathcal{C}):c\in\mathbf{E}} w_{\mathbf{E}} P(X_i|\mathbf{M}^*_{\mathbf{E}i}, \theta_{\mathbf{E}i})\right) \qquad (6.10)$$

The score improvement computation requires us to enumerate over all conditions in the condition set of interest and computationally is more expensive than the product model. The weights can also be estimated using an EM machinery of mixture of Gaussians, but we leave this for future work.

Structure learning approach for k graphs

Our score for structure learning is based on the pseudo likelihood of the data given model and requires us to compute the conditional probability distribution of each variable in a condition c. Assuming one of the conditional models above, our pseudo-likelihood score is defined as:

$$S(G_1, \cdots, G_k) = \sum_c \text{PLL}(D_c|\theta_c) + \text{MDL Penalty} \tag{6.11}$$

where

$$\text{PLL}(D_c|\theta_c) = \sum_j^{|D_c|} \sum_i^{N} \log P(X_i = x_{ij}|\mathbf{M}_{ci} = \mathbf{m}_{ci}, \theta_{ci}) \tag{6.12}$$

Note θ_{ci} is composed of parameters obtained from data in condition c as well as any subset of $\mathcal{C}$ that includes c and has an edge connected to X_i.

Our structure learning algorithm begins with k empty graphs and proposes edge additions for all variables, for all subsets of the condition set $\mathcal{C}$. The `while` loop iteratively makes edge modifications until the score no longer improves. The outermost `for` loop (Steps 4-17) iterates over variables X_i to identify new candidate MB variables X_j in a condition set $\mathbf{E}$. We iterate over all candidate MBs X_j (Steps 5-15) and condition sets $\mathbf{E}$ (Steps 6-14) and compute the score improvement for each pair $\{X_j, \mathbf{E}\}$ (Step 16). In Steps 7-9 we add a check that if a variable X_j is already present in any subset or superset $\mathbf{D}$ of $\mathbf{E}$, we do not include it as a candidate. This check prevents double counting of edge in overlapping condition sets. If the current condition set under consideration has more than one conditions, data from these conditions is pooled and parameters for the new distribution $P(X_i|\mathbf{M}^*_{\mathbf{E}i})$ is estimated using the pooled dataset (Steps 10-12). A candidate move for a variable X_i is composed of a pair $\{X'_j, \mathbf{E}'\}$ with the maximal score improvement over all variables and conditions (Step 16). After all candidate moves have been identified, we attempt all the moves in the order of decreasing score improvement (Step 18). Each move adds the edge $\{X_i, X'_j\}$ in condition set $\mathbf{E}'$. However, if either X_i or X'_j was already updated in a previous move, we ignore the move. Because not all candidate moves are made, by sorting the move order in decreasing score improvement, we enable moves with the highest score improvements to be attempted first. The algorithm converges when no edge addition improves the score of the k graphs.

Algorithm 3 NIPD

```
**Input:**
    Random variable set, $\mathbf{X} = \{X_1, \cdots, X_{|\mathbf{X}|}\}$
    Set of conditions $\mathcal{C}$
    Datasets of RV joint assignments, $\{D_1, \cdots, D_{|C|}\}$
**Output:**
    Inferred graphs $G_1, \cdots, G_{|C|}$
**while** Score$(G_1, \cdots, G_{|\mathcal{C}|})$ has not converged **do**
  **for** $X_i \in \mathbf{X}$ **do** {/**Propose moves**/}
    **for** $X_j \in (\mathbf{X} \setminus \{X_i\})$ **do**
      **for** $\mathbf{E} \in$ powerset$(\mathcal{C})$ **do**
        **if** $X_j \in \mathbf{M}^*_{i\mathbf{D}}$, s.t either $\mathbf{D} \subset \mathbf{E}$ or $\mathbf{E} \subset \mathbf{D}$ **then**
          Skip $X_j$.
        **end if**
        **if** $|\mathbf{E}| > 1$ **then**
          Estimate parameters for new conditional $P(X_i|\mathbf{M^*}_{\mathbf{E}i} \cup \{X_j\})$ using
          pooled dataset $\mathcal{D}_\mathbf{E}$ obtained from merging all $\mathcal{D}_e$ s.t. $e \in \mathbf{E}$.
        **end if**
        compute $\Delta Score_{\{X_i X_j\}\mathbf{E}}$.
      **end for**
    **end for**
    Store $\{X_i, X'_j, \mathbf{E}'\}$ as candidate move for $X_i$, where $\{X'_j, \mathbf{E}'\} =$
    $\arg\max_{j,\mathbf{E}} \Delta Score_{\{X_i X_j\}\mathbf{E}}$
  **end for**
  Make candidate moves $\{X_i, X'_j, \mathbf{E}'\}$ in order of decreasing score improvement /**Attempt moves to modify graph structures**/
**end while**
```

6.2.2 Experiments

We did some preliminary experimental analysis to compare our condition-specific learning approaches with the different parameterizations against the independent network learning approach. The independent network learning approach is referred to as INDEP.

Experimental setup: data description

We generated simulated datasets using two sets of networks of known structure, HIGHSIM and LOWSIM. All networks had the same number of nodes $n = 68$ and were obtained from the *E. coli* regulatory network [117]. To generate the network topologies we first obtained a sub network of $n = 68$ nodes, G_1, from the *E. coli* regulatory network. We then generated two networks, G_2 and G_3, by swapping 40% and 80% of G_1's edges, respectively while preserving the degree distribution of G_1. Swapping takes two edges $\{u, v\}$ and $\{x, y\}$, and replaces it by $\{u, x\}$ and $\{v, y\}$, where $u \neq v \neq x \neq y$. $\{G_1, G_2\}$ comprised HIGHSIM and $\{G_1, G_3\}$ comprised LOWSIM. For each pair of networks, we generated initial datasets using a differential equation-based gene regulatory network simulator [100]. We then split the data into two parts, learned two INDEP models for each partition, and generated data from these models. We repeated this procedure four times producing eight sets of simulated data with different parameters but the same network topology. It was possible to generate all eight sets from the regulatory network simulator by perturbing the kinetic constants, but our current data generation procedure was faster.

Results

Product model gets best structure especially in the face of limited training data We compared the overall structure of inferred networks using simulated data from two sets of networks, each set with two networks (Figs 6.3, 6.4, 6.5). In the first, HIGHSIM, the networks shared a larger portion (60%) of the edges, and in the second, LOWSIM, the networks shared a smaller (20%) portion of the edges. We compared the networks inferred by all the NIPD approaches to those inferred by INDEP by assessing the match between true and inferred node neighborhoods. Briefly, we split the data into p partitions, where $p \in \{2, 4, 6, 8, 10\}$, learned networks for each partition. The size of the training data decreased with increasing p. We obtained the number of nodes on which one approach was statistically significantly better (t-test p-value, < 0.05) in capturing its neighborhood, as a function of p.

We found that the NIPD-PROD approach is the best for getting the overall structure. The other NIPD approaches are beaten by INDEP. However, when we learn the weights (NIPD-WT-LEARN) as opposed to fixing them beforehand (NIPD-WTSUM), the performance margin from INDEP is smaller, indicating that learning weights is beneficial.

All NIPD models get better shared structure than INDEP We also compared the performance of the NIPD approaches against INDEP using the number of shared edges that were correctly captured (Fig 6.6). This is important because it is harder to get the similarities right especially in the case of limited data. We found the NIPD approaches get shared edges better than INDEP, regardless of the amount of sharing in the networks from each condition.

Discussion

Overall, we found that the NIPD-PROD approach was the best. It is surprising that this was the best because it was the simplest, and makes independence assumptions of the condition sets. The difference between the product and weighted sum models is that in the product model an edge needs to give improvement in the subset under consideration, whereas in the sum models the edge addition must have an improvement over all subsets. As a result the sum models may miss out on the weak shared edges because its contribution

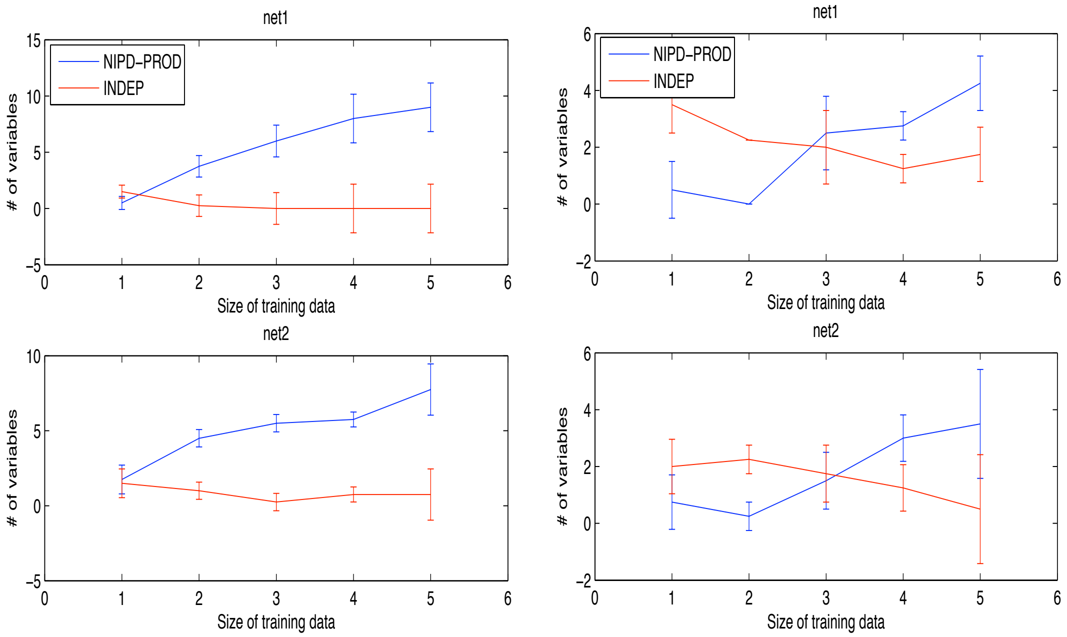


Figure 6.3: Structure comparison on the two sets of networks using the NIPD-PROD (NIPD with the product conditional) and INDEP models. The left column is for HIGHSIM and the right column is for the LOWSIM. x-axis corresponds to number of folds of training data. The size of training data in each fold decreases with the increasing fold count. y-axis is the number of variables on which one method is better than the other.

to the overall sum may be small. We attribute its good performance to its ability to capture those shared edges that may be missed because they are too weak to represent a benefit in the sum models. However, the fact that all models were able to get shared edges well in the face of limited training data is promising because finding shared patterns is harder under limited training datasets.

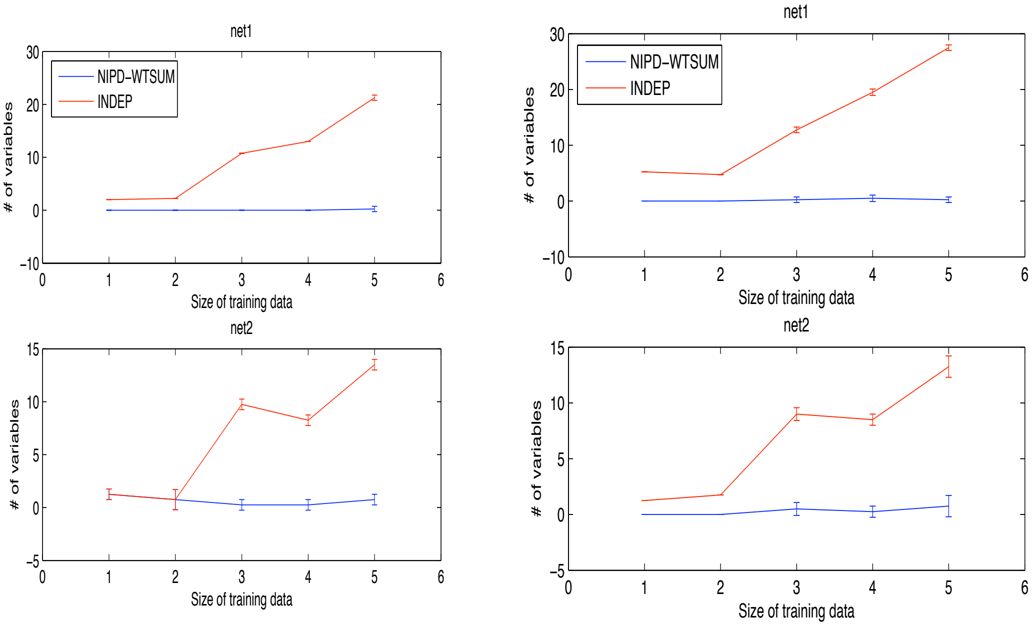


Figure 6.4: Structure comparison on the two sets of networks using the NIPD-WTSUM (NIPD with the weighted sum of conditionals) and INDEP models. The left column is for HIGHSIM and the right column is for the LOWSIM. x-axis corresponds to number of folds of training data. The size of training data in each fold decreases with the increasing fold count. y-axis is the number of variables on which one method is better than the other.

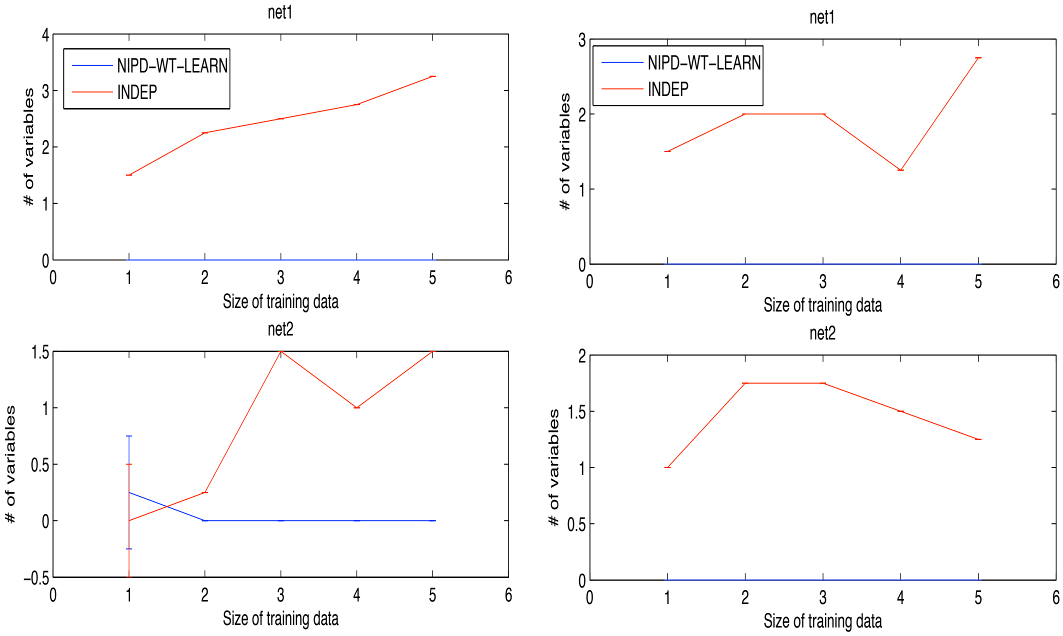


Figure 6.5: Structure comparison on the two sets of networks using the NIPD-WT-LEARN (NIPD with the weighted sum of conditionals with learned weights) and INDEP models. The left column is for HIGHSIM and the right column is for the LOWSIM. x-axis corresponds to number of folds of training data. The size of training data in each fold decreases with the increasing fold count. y-axis is the number of variables on which one method is better than the other. NIPD-WT-LEARN for LOWSIM and net1 of HIGHSIM is never able to outperform INDEP and therefore is on top on the x-axis.

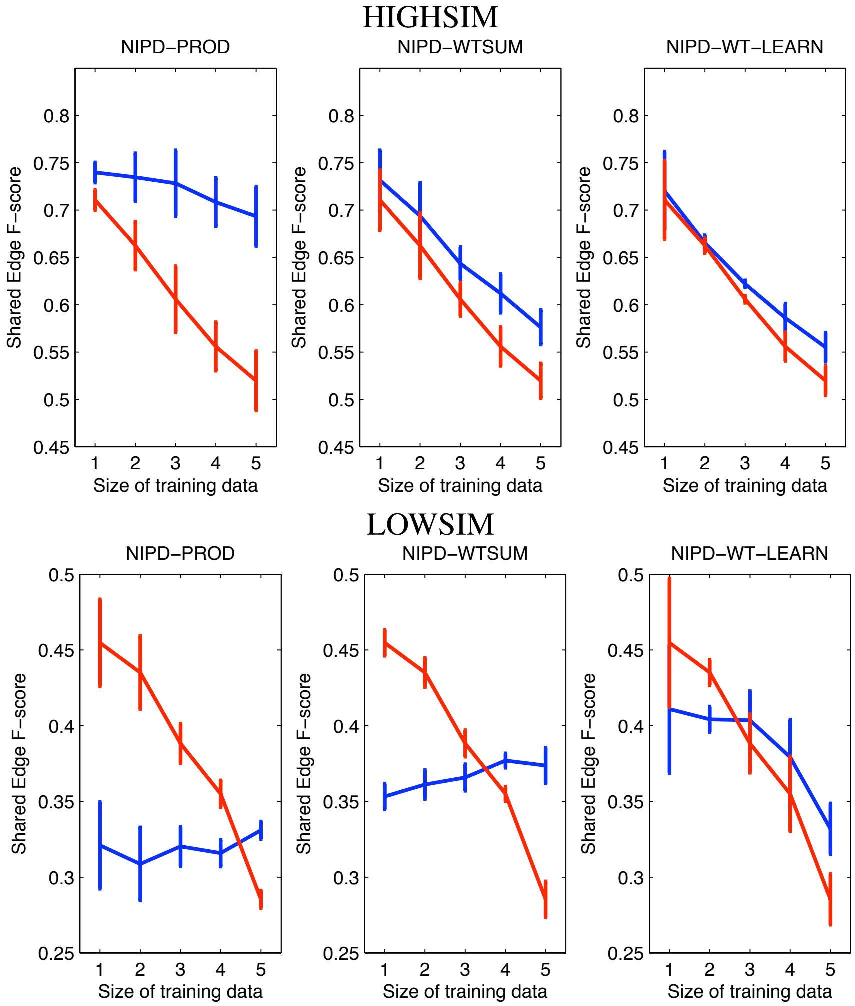


Figure 6.6: Comparison of different algorithms using match of shared edges. x-axis corresponds to number of folds of training data. The size of training data in each fold decreases with the increasing fold count. y-axis is the F-score of the shared edges in the true and inferred pairs of networks.

6.2.3 Learning condition-specific networks with unobserved condition

The above section assumed that the condition variable is known. However, in many real-world data situations we may not have the condition variable, because of two reasons: (a) dataset heterogeneity, and (b) to enable modeling complex sharing patterns. The first case arises when we are combining data from multiple related but not identical experimental setups, where it is difficult to ascertain the condition variable for each data point (joint assignment to all the random variables in our setup). An example of this scenario is combining expression data from different patients [74], or brain imaging data from the neuroscience domain [71].

The second case, where we deliberately choose to not enforce the condition variable assignment to a data point, gives us more flexibility to model different types of sharing patterns. Such complex sharing patterns can arise in biological networks because of modularity and component re-use [75, 101]. For example, master regulators such as *Twist* and *Mef2* combinatorially regulate overlapping sets of genes during different stages of fly muscle development [118]. More examples of such *cis-regulatory circuits* can be found in the yeast transcriptional network [61], and other developmental networks [33, 23]. From a modeling perspective, these sharing patterns may cause a data point to be associated with multiple hidden variables. This kind of flexibility is not available if we have a single, observed condition variable for the entire data point.

Whether it be by necessity or by choice, it is clear that we need models that can infer the condition variable during structure learning. We now describe different models that capture different types of sharing and specificity. We also address the question of parameter tying to incorporate sharing within the parameter structure of the conditional distributions. We first begin with the simplest model, mixture of graphs model, in which there is a single, hidden condition variable. We then consider the other extreme, with a different condition variable for each random variable, and then finally we consider the *middle-ground* model, where we have one variable for every graph component.

Modeling condition-specificity at different levels of granularity: An example

Before proceeding with the technical details of the different models we provide an overview of the basic differences in the different models (Fig 6.7). We assume that for each data point, the condition variable is not known and must be inferred. We know the number of conditions and our goal is to learn a network for each condition. However the models differ in the extent to which different parts of the network are controlled by a condition variable. We consider condition-specificity at level of the entire graph, at the level of individual variables and at the level of graph components. The corresponding models are the Mixture of Undirected Graphs (MUGs), Per-variable Expectation Maximization (PVEM) and the Graph Component with CONDitional model (GC-COND).

In Fig 6.7 we highlight the main differences between the three models. We use z (with or without subscript) to denote the condition variable. The value of z determines which set of neighbors must be used to generate the sample of a variable in a data point. This decision of selecting the value of z can be made for different parts of the network. We consider the case of two conditions $\mathcal{C} = \{1, 2\}$, where 1 is denoted by red and 2 by green. The example graph has five variables $\{X_1, \cdots, X_6\}$. The red and green edges specify the connections in the different condition-specific graphs, that is, all the red edges specify the graph for condition 1 and all the green edges specify the connections in condition 2. The assignment to the z variable determines which graph the sample for a random variable is generated. The union of all these edges is called the *union graph*.

In the MUG model, there is a single z variable associated with the entire graph structure. The decision about which condition a sample is from is made for all the random variables simultaneously by the assignment of the z variable. While this is a simple and intuitive model, it does not allow for fine-grained control of condition-specificity for different sets of variables. For example, we may have a situation where some genes are more shared and some genes are more specific to each condition. By having a single variable for all variables, we cannot incorporate this information.

In the PVEM model, every random variable has its own condition variable z_i. The condition from which a sample is generated is selected for each variable independently. This is the most general type of model, allowing each variable to have its own level of sharing between the different conditions. However, while this model is the most general, in order to make it tractable to learn we must infer the condition variables independently for each random variable. This causes some consistency violations in the inference of the condition variables. The last kind of model which takes care of the consistency problem and also allows sufficient granularity, is the GC-COND model. In this model, we have a condition variable for each connected component of the union graph. The example in Fig 6.7 has two components and therefore there are two condition variables z_1, z_2.

Mixture of (non-decomposable) undirected graphs

The mixture of undirected graphs (MUG) model is a mixture model for defining probability distribution over a high-dimensional random variable [62, 99, 122, 35]. This high-dimensional random variable is the set $\mathbf{X}$ of random variables $X_1, \cdots, X_N$, where each X_i itself is representing a gene. The MUG model is similar to a mixture of directed acyclic graphs (DAGs) [134], with the exception that we learn undirected graphs. As described above, we have data from K conditions, but we do not know the condition labels of each dataset. In our gene expression case each data point is a microarray measuring the expression of N genes. The generative model assumes that the data is generated as follows: there is a hidden stochastic process that randomly selects an assignment to the condition variable and thus a graph model, which in turn determines the gene expression of all genes in that condition, and then generates samples for all the genes using the selected model.

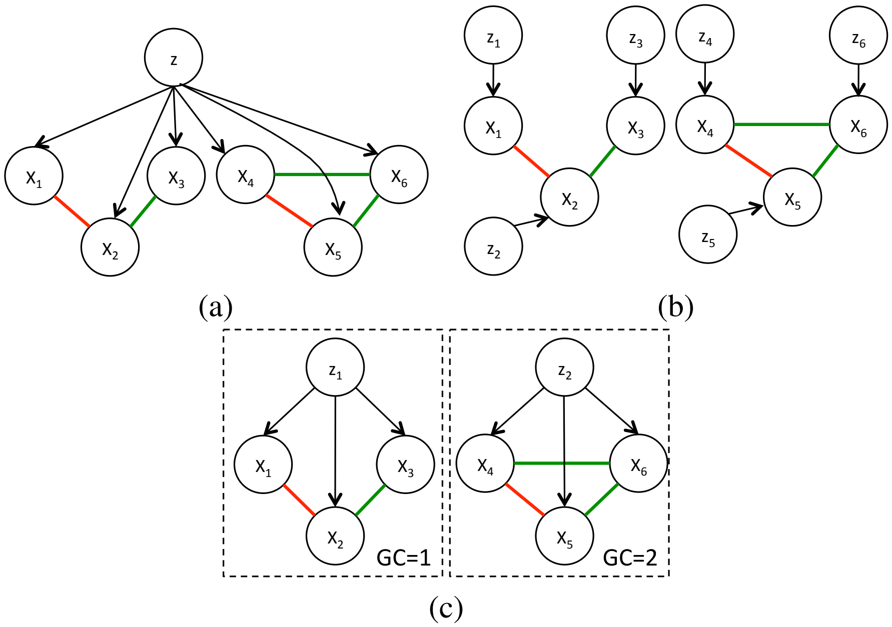


Figure 6.7: Mixture models controlling condition-specificity at different levels of granularity. (a) MUGs, (b) PVEM, (c) GC-COND. The colored undirected edges distinguish the networks for each condition: red for one condition and green for another condition.

MUG in detail Let $\mathbf{X} = \{X_1, \cdots, X_N\}$ represent the set of observed random variables. Each joint assignment $\mathbf{x}$ to $\mathbf{X}$ is generated using a mixture of K undirected graphs as follows:

- $k \sim$ Multinomial$(\alpha_1, \cdots, \alpha_F)$
- $\mathbf{x} \sim P(X_1, \cdots, X_N | G_k, \theta_k)$, where G_k is the graph structure and θ_k is the set of parameters corresponding to the k^{th} mixture component.

The joint probability distribution of this mixture model is

$$P(\mathbf{X} = \mathbf{x}) = \sum_{k=1}^{K} \alpha_k P(\mathbf{x} | G_k, \theta_k) \tag{6.13}$$

We rewrite the full joint as product of local conditionals

$$P(\mathbf{X} = \mathbf{x} | G_k, \theta_k) = \prod_{i=1}^{N} P(X_i = x_i | \mathbf{M}_{ki} = \mathbf{m}_{ki}, \theta_{ki}).$$

If each G_k is a Bayesian network, $\mathbf{M}_{ki}$ are the set of parents of X_i in G_k and the product yields the likelihood of the data given model. However, in our case G_k is an undirected graph $\mathbf{M}_{ki}$ is the Markov blanket of X_i and the product yields the pseudo likelihood of the data given model. As described in the previous chapters, this is a commonly used approximation and enables us to perform search in a tractable fashion without having to compute the partition function [64, 105].

The goal of structure learning is to find the best $\mathcal{G} = \{G_1, \cdots, G_K\}$ and $\theta_1, \cdots, \theta_K$ such that likelihood of data $\mathbf{D} = \{D_1, \cdots, D_K\}$ given this mixture of graphs is maximized. That is,

$$(\mathcal{G}^*, \theta_1^*, \cdots, \theta_K^*) = \arg\max_{\mathcal{G}} \max_{\theta_1, \cdots, \theta_K} P(\mathbf{D}|\mathcal{G}, \theta_1, \cdots, \theta_K) \tag{6.14}$$

where $\mathbf{D}$ is our data. Assuming the data are independently and identically distributed (IID) and using the definition of pseudo likelihood we have

$$\log P(\mathbf{D}|\mathcal{G}, \theta_1, \cdots, \theta_K) = \sum_{d=1}^{|\mathbf{D}|} \log \left(\sum_{k=1}^{K} \alpha_k \left(\prod_{i=1}^{N} P(X_i = x_{di}|\mathbf{M}_{ki} = \mathbf{m}_{kdi}, \theta_{ki}) \right) \right) \tag{6.15}$$

We refer to the left hand side as $PLL(\mathbf{D})$. The problem here is the sum inside the log, which prevents the objective from decomposing into anything tractable. To get around this problem we use the typical Expectation Maximization (EM) framework where we augment each $\mathbf{x}_d$ with an additional dimension z_d which stores the mixture component from which $\mathbf{x}_d$ was generated. We let $\mathbf{z} = \{z_1, \cdots, z_d\}$, $\mathbf{Z} = \{Z_1, \cdots, Z_{|\mathbf{D}|}\}$. Assuming $G_1, \cdots, G_k$ are fixed, we only need to estimate the parameters $\theta_1, \cdots, \theta_K$. We refer to these parameters collectively as Θ. Now we write the expected pseudo likelihood of the completed data as follows, where the expectation is w.r.t to the conditional distribution $P(\mathbf{Z}|\mathbf{D}, \Theta')$, where Θ' is the current estimate of parameters (from a prior or previous iteration) :

$$E_{P(\mathbf{Z}|\mathbf{D},\theta')}[PLL(\Theta;\mathbf{z},\mathbf{D})] = \sum_{\mathbf{z}\in\mathcal{Z}} P(\mathbf{Z}|\mathbf{D},\Theta')\log P(\mathbf{Z},\mathbf{D}|\Theta) \tag{6.16}$$

Here $\mathcal{Z}$ is the set of possible assignments to $\mathbf{Z}$. We rewrite the above as

$$Q(\Theta,\Theta') = \sum_{\mathbf{z}\in\mathcal{Z}} P(\mathbf{Z}|\mathbf{D},\Theta')\log P(\mathbf{D}|\mathbf{Z},\Theta)P(\mathbf{Z}|\Theta) \tag{6.17}$$

After doing some algebra (Appendix C) we derive a decomposable pseudo likelihood score which decomposes into pseudo likelihood scores of a variable and its neighborhood:

$$Q(\Theta,\Theta') = \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{d=1}^{|\mathbf{D}|} \gamma_{dk}\log\alpha_k^{\frac{1}{N}} P(X_i = x_{id}|\mathbf{M}_{ki} = \mathbf{m}_{kdi}, \theta_{ki}) \tag{6.18}$$

The sum decomposes over each variable, indicating we can optimize this sum by independently optimizing over each variable.

Unfortunately, this alone does not help during structure search. This is because the γ's are computed with respect to the current network set $\mathcal{G}$ and parameters Θ'. When we change any network of any component by, say, adding an edge, the γ's would need to be recomputed, and as a result we would need to re-estimate the parameters of all variables, even if they are not getting modified structurally. Therefore, we make another approximation. When we consider the addition of a new edge, $\{X_i, X_j\}$ to model l, we use the γ's from the previous iteration and re-estimate parameters for the proposed edge using these old γ's. Then we identify the best set of moves (edge additions) using the score improvement, $\Delta S = \Delta S_{X_i} + \Delta S_{X_j}$ where

$$\begin{aligned}\Delta S_{X_i} = {} & \sum_{k=1}^{K}\sum_{d=1}^{|\mathbf{D}|} \gamma_{dk}\log\alpha_k^{\frac{1}{N}} P(X_i = x_{id}|\mathbf{M}_{ki} \cup \{X_j\} = \mathbf{m}_{kid} \cup \{x_{kjd}\}, \theta'_{ki}) \\ & - \sum_{k=1}^{K}\sum_{d=1}^{|\mathbf{D}|} \gamma_{dk}\log\alpha_k^{\frac{1}{N}} P(X_i = x_{id}|\mathbf{M}_{ki} = \mathbf{m}_{kid}, \theta_{ki})\end{aligned} \tag{6.19}$$

Here $\theta'_{ki} = \theta_{ki}$ if $k \neq l$. Otherwise θ_{li} is re-estimated after adding the new neighbor X_j to X_i's neighborhood in graph G_l. That is, we re-use all parameters other than the parameters of the conditional associated with X_i.

We now describe the algorithm steps (Algorithm 4). In the top `for` loop we assume the γ_{kid} are fixed and search for the next possible Markov blanket variable for a variable X_i. Note this search happens over all the model graphs and produces a pair $\{X_j, l\}$ specifying the next Markov blanket addition in the graph G_l. We also impose a maximum size of the Markov blanket of each variable in each condition. The moves are then made in increasing order of the score improvement such that the moves that have the maximal score improvement are attempted first. Note that a proposed move becomes invalid if either X_i or X_j has been modified in any of the graphs in a preceding move. After all the graph modifications have been done, we fix the graph structure and do EM-based parameter estimation until the score does not change. Because the conditionals take the form of conditional Gaussians, the update equations for γ_{kid} and the parameters all use the standard multi-variate Gaussian mixture model framework. During the maximization step we take the graph structure into account while estimating the co-variance matrix for the Gaussian. We also smooth the γ_{kid} with a small correction term $\epsilon =$ 1E-5 to avoid data stealing.

Per-variable EM (PVEM) model

In this approach we add more flexibility to the model by allowing each variable X_i to have a separate condition variable. This model is similar to modeling different interventional distributions [38], where different parameters are selected based on the assignment of the intervened variable. However, we handle a more general case where the condition variable influences the neighborhood structure, and therefore also the parameters. Each condition variable determines the set of neighbors that influence a random variable at any point in time. The d^{th} sample is generated is as follows: For each X_i

- $k \sim$ Multinomial$(\alpha_1, \cdots, \alpha_k)$
- $x_{id} \sim P(X_i|\mathbf{M}_{ki} = \mathbf{m}_{kid})$

Algorithm 4 MUG structure learning

```
Input:
  Random variable set, $\mathbf{X} = \{X_1, \cdots, X_{|\mathbf{X}|}\}$
  Dataset of RV joint assignments, $\mathbf{D} = \{D_1, \cdots, D_K\}$
Output:
  Inferred graphs $\mathcal{G} = \{G_1, \cdots, G_k\}$
while Score not converged do
   /*Modify graph structure*/
   for $X_i \in \mathbf{X}$ do
      Find best new pair $\{X_j, l\}$, $1 \leq l \leq K$ that maximizes $\Delta S_{ij}$ using Eq 6.19
   end for
   Update graphs by adding edges in decreasing order of $\Delta S_{ij}$.
   /*EM on parameters*/
   while Score not converged do
      Expectation step to re-estimate $\gamma_{kid}, 1 \leq k \leq K, 1 \leq i \leq N, 1 \leq d \leq |\mathbf{D}|$
      Maximization step to re-estimate $\alpha_k$ and parameters of the conditionals $P(X_i|\mathbf{M}_{ki})$ using the new $\gamma_{kid}$'s
   end while
end while
```

where $\mathbf{M}_{ik}$ is the MB of X_i in condition k, $\mathbf{m}_{kid}$ is the assignment to $\mathbf{M}_{ki}$ in the d^{th} data point[3].

We apply the pseudo likelihood decomposition before applying the EM framework

$$\log P(\mathbf{D}|\mathcal{G}, \theta_1, \cdots, \theta_k) = \sum_{d=1}^{|\mathbf{D}|} \sum_{i=1}^{N} \log P(X_i = x_{di}|\mathbf{M}_i = \mathbf{m}_{di}) \tag{6.20}$$

[3] This requires that we already have the assignment for all the variables in $\mathbf{M}_{ik}$, which may not be true depending on the ordering of the variables during sampling. However, data is generated using a Gibbs sampling framework where a variable $X_j \in \mathbf{M}_{ik}$ may have the assignment from the previous iteration or the current iteration.

Here $\mathbf{M}_i = \cup_{k=1}^{K}\mathbf{M}_{ki}$, where $\mathbf{M}_{ki}$ are the set of neighbors of X_i in graph G_k. Assuming a conditional Gaussian for each conditional distribution, we have a mixture of conditional Gaussians for each variable X_i. The k^{th} component of the mixture describes a conditional Gaussian density $P(X_i|\mathbf{M}_{ki} = \mathbf{m}_{kid})$:

$$P(X_i|\mathbf{M}_{ki} = \mathbf{m}_{kid}) = \mathcal{N}(\mathbf{A}_{ki}\mathbf{m}_{kid} + b_{ki}, \sigma_{ki})$$

Here $\mathbf{A}_{ki}$ is a vector of coefficients for the linear combination of the neighbor set values, and σ_{ki} is the conditional variance of X_i. It is assumed to be the same for all joint assignments of $\mathbf{M}_{ki}$. The complete pseudo likelihood score of the graphs is

$$\log P(\mathbf{D}|\mathcal{G}, \theta_1, \cdots, \theta_k) = \sum_{d=1}^{|\mathbf{D}|}\sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\alpha_{ki}\mathcal{N}(\mathbf{A}_{ki}\mathbf{m}_{kid} + b_{ki}, \sigma_{ki})\right) \tag{6.21}$$

where $\mathbf{M}_i = \bigcup_k \mathbf{M}_{ki}$ and α_{ki} are the standard mixing weights for each component.

We now need to do EM for every variable X_i. We introduce the typical hidden variables z_{kid} specifying the model from which the assignment to X_i is obtained in datapoint d. γ_{kid} specifies the expected value of $z_{kid} = 1$, that is, generating X_i's value in the d^{th} datapoint from the k^{th} mixture component. The mixture of conditional Gaussians for each variable is given by:

$$P(X_i = x_{id}|\mathbf{M}_i = \mathbf{m}_{id}) = \sum_{k=1}^{K}\alpha_{ik}\frac{1}{\sqrt{2\pi\sigma_{ik}}}exp\left(\frac{(x_{id} - \mathbf{A}_{ki}\mathbf{m}_{kid} - b_{ki})^2}{-2\sigma_{ki}}\right) \tag{6.22}$$

Applying the EM crank the expected pseudo-likelihood for each variable is:

$$Q(\theta_{ki}, \theta'_{ki}) = \sum_{d=1}^{|\mathcal{D}|}\sum_{k}\gamma_{kid}\left(\log\alpha_{ki}\frac{1}{\sqrt{2\pi\sigma_{ki}}} - \frac{1}{2\sigma_{ki}}(x_{id} - \mathbf{A}_{ki}\mathbf{m}_{kid} - b_{ki})^2\right) \tag{6.23}$$

The update equations for the different parameters σ_{ki}, $\mathbf{A}_{ki}$ and b_{ki} can be obtained by deriving $Q(\theta_{ki}, \theta'_{ki})$ with respect to each parameter, setting to 0 and solving.

1. **ML estimate of σ_{ki}**

$$\sigma_{ki} = \frac{\sum_{d=1}^{|\mathcal{D}|}\gamma_{kid}(x_{id} - \mathbf{A}_{ki}\mathbf{m}_{kid} - b_{ki})^2}{\sum_{d=1}^{|\mathcal{D}|}\gamma_{kid}}$$

2. **ML estimate of** A_{ki}

$$\mathbf{A}_{ki} = \frac{\sum_{d=1}^{|\mathcal{D}|} \gamma_{kid}(x_{id} - b_{ki})\mathbf{m}_{kid}^{\mathsf{T}}}{\sum_{d=1}^{|\mathcal{D}|} \gamma_{kid}\mathbf{m}_{kid}\mathbf{m}_{kid}^{\mathsf{T}}} \tag{6.24}$$

Structure learning in this model is much harder than the MUG model because of the presence of more hidden variables (Algorithm 5). However, unlike the MUG model where scoring a single move would have required us to re-estimate all the parameters of the conditionals because of a single γ_{kd} for all random variables, in this case we can re-compute the γ_{kid} for each proposed move for each conditional independently. For every candidate neighbor we do an EM run for estimating the parameters given the proposed structure (Algorithm 6 for `scoreMoveEM`). Since every step of EM is guaranteed to improve the likelihood, we need not run EM to convergence, but to a fixed q number of steps. We then compare the candidate structures based on the score computed from parameters after q EM steps. The overall structure learning algorithm is similar to the original MBS algorithm. The main difference is the introduction of an EM sub-routine to score candidate moves.

MUG per connected component

The standard mixture of undirected graph model imposes a single condition variable for all the random variables. In contrast, the PVEM model has a condition variable for each random variable. While the PVEM model gives the maximum amount of flexibility, it violates some consistency in the assignments of the condition of a variable and of the conditions of the Markov blanket variables. This is because the condition variable is inferred for each random variable independently. The constraint that we must impose is that if a variable X_j in a datapoint d is being generated from model l, then to guarantee that the dependency between X_j and another variable X_i is maintained in condition l, X_i too must be generated from the l^{th} model in X_i's mixture. However this constraint between the different condition variables prevents us from inferring the condition variables independently. This

Algorithm 5 PVEM structure learning

```
Input:
  Random variable set, $\mathbf{X} = \{X_1, \cdots, X_{|\mathbf{X}|}\}$
  Dataset of RV joint assignments, $\mathbf{D} = \{D_1, \cdots, D_K\}$
Output:
  Inferred graphs $\mathcal{G} = \{G_1, \cdots, G_k\}$
while Score not converged do
  for $X_i \in \mathbf{X}$ do
    for $X_j \in \mathbf{X}$ and $X_j \neq X_i$ do
      for $l \in \mathcal{C}$ do
        $\Delta S_{X_i} = \texttt{scoreMoveEM}(X_j, l, X_i)$
        $\Delta S_{X_j} = \texttt{scoreMoveEM}(X_i, l, X_j)$
        $\Delta S_{\{X_i,X_j\},l} = \Delta S_{X_i} + \Delta S_{X_j}$
      end for
    end for
    $\{X_j, l\} = \arg\max_{X'_j \neq X_i, l'} \Delta S_{\{X_i,X'_j\},l'}$
  end for
  Update graphs by adding edges in decreasing order of score improvement.
end while
```

problem is much harder to solve because now the dimensionality of the hidden variables corresponding to the conditions increases exponentially because we have to jointly infer the condition variables. We instead propose a simpler solution, which is based on keeping the number of condition variables flexible during structure learning. In particular we will have a hidden condition variable for each connected component of the *union graph*. Because we are doing structure learning, the graph structure changes, and so does the number of connected components.

Our model now has MUG model for every connected component. Let $\widehat{G} = \cup_k G_k$ represent the union graph that can exist in any of the K conditions. Let $CC_{\widehat{G}}$ denote the

Algorithm 6 Sub-routine `scoreMoveEM`

```
Input:
  Variable $X_i$ and candidate MB variable $X_j$
  Condition variable $l$
Output:
  Score improvement if $X_j$ were added to $X_i$'s MB in condition $l$
Initialize parameters $\theta_{ijl}$ for $P(X_i|\mathbf{M}_{li} \cup \{X_j\})$
$iter = 0$
repeat
  Estimate $\gamma_{ijk}$ $\forall k$ using existing $\theta_{ijl'}, l' \neq l$ and the new $\theta_{ijl}$
  Estimate $\alpha_{ik}$ $\forall k$ using new $\gamma_{ijk}$'s
  Compute pseudo likelihood score for $P(X_i|\mathbf{M}_{il} \cup \{X_j\})$
  $iter++$
  Update all parameters $\theta_{ijk}$ $\forall k$.
until $iter = q$ or pseudo likelihood score does not change
```

connected components of $\widehat{G}$. Now the data is generated is as follows:

- For each $G_c \in CC_{\widehat{G}}$, $k \sim$ Multinomial$(\alpha_1, \cdots, \alpha_K)$
- For each $X_i \in$ VertexSet(G_c), $X_i \sim P(X_i|\mathbf{M}_{ki})$

The joint probability distribution can be written as

$$P(\mathbf{X} = \mathbf{x}) = \prod_{c=1}^{|CC_{\widehat{G}}|} P(\mathbf{X}_c = \mathbf{x}_c) \tag{6.25}$$

where $\mathbf{X}_c$ is the vertex set of c^{th} connected component of $\widehat{G}$. We now have a MUG model for every connected component:

$$P(\mathbf{X} = \mathbf{x}) = \prod_{c=1}^{|CC_{\widehat{G}}|} \sum_{k=1}^{K} \alpha_{ck} P(\mathbf{X}_c = \mathbf{x}_c|\theta_{ck}) \tag{6.26}$$

the inner joint can be written as a product of conditionals if we use pseudolikelihood:

$$P(\mathbf{X}=\mathbf{x}) = \prod_{c=1}^{|CC_{\widehat{G}}|} \sum_{k=1}^{K} \alpha_{ck} \prod_{i=1}^{|\mathbf{X}_c|} P(X_i = x_i|\mathbf{M}_{ki} = \mathbf{m}_{ki}, \theta_{ki}) \tag{6.27}$$

We introduce a hidden variable for each data point, d and each connected component c, Z_{dc}. Applying the EM crank for each MUG for each components produces the following decomposition:

$$Q(\Theta,\Theta') = \sum_{c=1}^{|CC_{\widehat{G}}|} \sum_{i=1}^{|\mathbf{X}_c|} \sum_{d=1}^{|\mathbf{D}|} \sum_{k=1}^{K} \gamma_{cdk} log \alpha_{ck}^{\frac{1}{|\mathbf{X}_c|}} P(X_i = x_{di}|\mathbf{M}_{ki} = \mathbf{m}_{kdi}, \theta_{ki}) \tag{6.28}$$

Structure learning of this model is again harder than the MUG model. The problem is because of not knowing the structure $\widehat{G}$, and therefore the connected components of $\widehat{G}$ either. Currently, during learning a single MUG model, we make use of the γ's from the previous E step to estimate score of the new model. The old γ's are also used to estimate the parameters of the new conditional distribution. But now since each datapoint could have potentially many γ's, one for each connected component, we need to decide which one to use during network scoring and parameter estimation of the new conditional. This issue can be addressed in two ways, which we describe below.

One way is to re-estimate the γ's every time two different graph components can merge. For example, say X_i currently belongs to component G_p and X_j belongs to component G_q. To compute the score improvement for the addition of the edge $\{X_i, X_j\}$, we first merge G_p and G_q into $\mathbf{C}_r$, erase all γ_{pdk} and γ_{qdk} and re-estimate a new γ_{rdk} for this new component and estimate the score on this component. Initially, when the graph is completely disconnected, we would have to re-estimate γ's for almost every proposed structure modification. However, as the graph would get more connected, the number of re-estimations would get reduced.

Another more efficient way is to approximate new γ's by taking a weighted mean of the original γ's from component G_p and component G_q, and then estimate the parameters of the conditional using the new estimated γ's. For example in the above example $\gamma_{rdk} = w_p\gamma_{pdk} + w_q\gamma_{qdk}$. This is more computationally efficient as we do not reestimate the γ's at all. We can set the weights to be proportional to the contribution of component to the overall pseudo likelihood, $w_p \propto$ PLL$(VertexSet(G_p))$, or even to the size the of each component. In our implementation we briefly experimented with both approaches and found the weighted approach, with weights proportional to connected component size, to be much better in terms of inferred structure quality.

In general, before we compute the score improvement for a new edge, we need to make sure that we have graph component for the edge. Assuming that the new edge will affect or create component G_r, the score improvement is given by a similar formulation as in the previous section: $\Delta S_{\{X_i,X_j\},l} = \Delta S_{X_i} + \Delta S_{X_j}$, where

$$\begin{aligned}\Delta S_{X_i} = &\sum_{k=1}^{K}\sum_{d=1}^{D}\gamma_{rdk}\text{log}\alpha_{rk}^{\frac{1}{|G_r|}}P(x_{di}|\mathbf{m}_{kdi}\cup\{x_{dj}\},\theta'_{ki}) \\ &-\sum_{k=1}^{K}\sum_{d=1}^{D}\gamma_{rdk}\text{log}\alpha_{rk}^{\frac{1}{|G_r|}}P(x_{di}|\mathbf{m}_{kdi}\cup\{x_{dj}\},\theta_{ki}) \qquad (6.29)\end{aligned}$$

The overall structure learning algorithm is similar to the MUG model and described in Algorithm 7.

GC-COND with parameter tying

Sharing of information across conditions can also be done by parameter tying of the conditional distributions [41, 125]. The parameters that are tied or shared across conditions are estimated jointly from the different conditions and exploit the shared information more efficiently. This also reduces the number of parameters that are associated with the model. We now describe how to implement parameter tying using the conditional Gaussian framework. To explicitly represent the shared neighbor in the parameters of our model, we force the dimension of the weight vectors, $\mathbf{A}_{ki}$ corresponding to the shared neighbor to have the same value. However, because of this sharing constraint, the ML estimates of each $\mathbf{A}_{ki}$, cannot be derived independently. To incorporate this sharing constraint, we split the weight vector into the shared and unshared parts, and take derivatives with respective to these parts. Let $\mathbf{A}_{ki}$ represent the weight vector for variable X_i in condition k. We write $\mathbf{A}_k = [\mathbf{B}_{ki}\mathbf{C}_i]$, where $\mathbf{B}_{ki}$ represents the weights for neighbors of X_i specific to condition k, and $\mathbf{C}$ represent the weight vector for the shared neighbors across conditions. Similarly we write $\mathbf{m}_{kid}$, which is the joint assignment to neighbors of X_i in condition k from datapoint d as, $\mathbf{m}_{kid} = [\mathbf{p}_{kid}\mathbf{q}_{id}]$, where $\mathbf{p}_{kid}$ represents the assignment to the condition k-specific neighbors of X_i and $\mathbf{q}_{id}$ represents the assignment to the shared neighbors of X_i. We provide the equation for the two condition case $\mathcal{C} = \{1, 2\}$ and show details in Appendix D. The general form can be derived from here.

- **ML estimate of shared parameters** $\mathbf{C}$ We assume X_i belongs to the c^{th} graph component. Therefore all the γ's associated with X_i are γ_{ckd}.

$$\mathbf{C} = \frac{\sum_{d=1}^{|\mathcal{D}|} \gamma_{c1d} \left(\frac{(x_{id} - \mathbf{B}_{1i}\mathbf{p}_{1id} - b_{1i})\mathbf{q}_{id}^{\mathsf{T}}}{\sigma_{1i}} \right) + \gamma_{c2d} \left(\frac{(x_{id} - \mathbf{B}_{2i}\mathbf{p}_{2id} - b_{2i})\mathbf{q}_{id}^{\mathsf{T}}}{\sigma_{2i}} \right)}{\sum_{d=1}^{|\mathcal{D}|} \frac{\gamma_{c1d}\mathbf{q}_{id}\mathbf{q}_{id}^{\mathsf{T}}}{\sigma_{1i}} + \frac{\gamma_{c2d}\mathbf{q}_{id}\mathbf{q}_{id}^{\mathsf{T}}}{\sigma_{2i}}}$$

- **ML estimate of unshared parameters** B_k

$$\mathbf{B}_k = \frac{\sum_{d=1}^{|\mathcal{D}|} \gamma_{ckd}(x_{id} - \mathbf{C}\mathbf{q}_{id} - b_{ki})(\mathbf{p}_{kid}^{\mathsf{T}})}{\sum_{d=1}^{|\mathcal{D}|} \gamma_{ckd}\mathbf{p}_{kid}\mathbf{p}_{kid}^{\mathsf{T}}}$$

6.3 Experimental setup

We had three main goals of our experiments: (a) assess the model performance when the model generating the data is different from the model being learned, (b) assess the effect of union graph topology on network structure inference, (c) assess the benefit of parameter tying. We execute all experiments on simulated data using networks of known ground truth. For (a) we consider three pairs of networks, NET12, NET16 and NET12-66

(Fig 6.8). These three pairs of networks have different topologies of the underlying union graph. Using each pair of networks we generate data using three generative models: (a) Constrained (CONSTR), (b) MUG and (c) Independent (INDEP). The CONSTR generative model uses a PVEM generative model but makes sure that during data generation the condition assignments of variables are consistent within a single data point. The MUG model is the mixture of undirected graphs and the INDEP model just generates two independent datasets for the two networks. We first compare the performance of PVEM, MUGs, INDEP, GC-COND models on the CONSTR datasets. We then compare the performance of MUG, GC-COND and INDEP models on the MUG and INDEP datasets.

For (b) we consider an additional network pair NET12-79 as this together with the existing three pairs of networks gives us a range of networks with different topologies of the union graph. We examine the PVEM, GC-COND and MUG models on the CONSTR datasets.

For (c) we consider only the CONSTR datasets but on several network pairs, NET12, NET13, NET14 and NET16. These networks have different topologies and vary in the number of shared edges (Table 6.8). We compare parameter tying versus no tying in the GC-COND model. The model with parameter tying is referred to as GC-CONDSH. Our complete experimental design is described in Table 6.7.

6.4 Results

6.4.1 Generative model comparison

We first compared the performance of all four models (INDEP, MUGs, PVEM, GC-COND) on the three networks pairs NET12, NET16 and NET12-66 with data from the CONSTR model (Figs 6.9,6.11,6.13, summarized in Table 6.9). All models outperformed

Experiment	Network pairs	Generative Model	Test Models
Model assumption	NET12 NET16 NET12-66	CONSTR, MUGs, INDEP	PVEM,MUGs, GC-COND,INDEP
Topology of union graph	NET12 NET16 NET12-66 NET12-79	CONSTR	PVEM,MUGs, GC-COND
Parameter tying	NET12, NET13 NET14, NET16	CONSTR	GC-COND, GC-CONDSH

Table 6.7: Experimental design to analyze the different questions. Generative models are used to generate the data and test models are learned.

Network Pair	Node	NET1 Edge Cnt	NET2 Edge Cnt	Shared Edge Cnt	% of shared edges
NET12	68	49	49	29	60%
NET13	98	79	81	38	47%
NET14	99	79	82	24	29%
NET16	68	51	55	11	20%

Table 6.8: Number of shared edges in each network pair. The percentage of shared edges is the smallest of the two networks.

the INDEP model, especially at small training dataset sizes, suggesting that when the data is mixed, a model which assumes the label to be given has poor performance. Comparison of the other models showed surprising results. We expected that the PVEM model would have the best performance since the generative model was closest to PVEM. However, we found that the MUGs model performs the best followed by the GC-COND, and GC-COND occasionally outperformed the PVEM model. One reason why this might be happening is that structure of the union graph imposes constraints on the condition labels, which is automatically imposed in the MUGs model. In contrast the GC-COND model must infer this and the PVEM model does not even impose this constraint. The learning task for GC-COND and PVEM is much harder than MUGs model.

We then compared the performance of three models (INDEP, MUGs, GC-COND) on

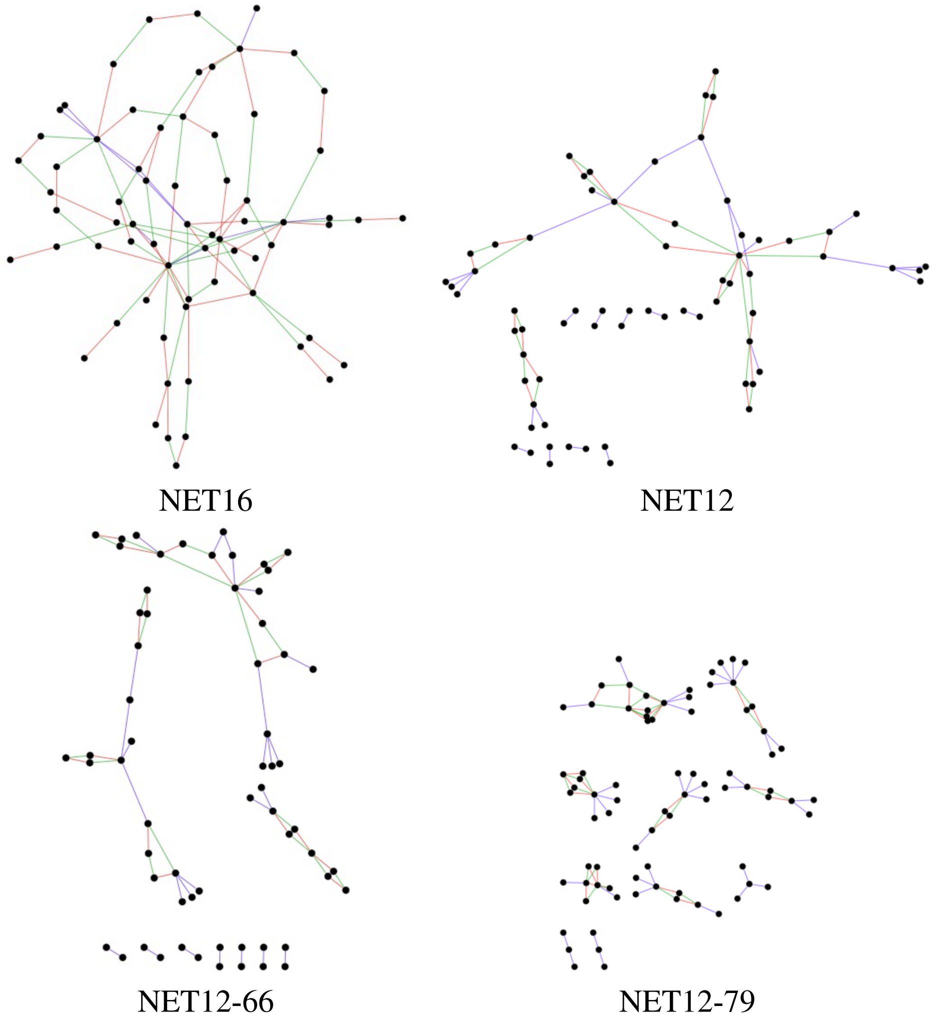


Figure 6.8: Union networks for the network pairs used in the simulations. The edge colors indicate the condition-specificity, with red for condition 1, green for condition 2 and blue denoting a shared edge between conditions 1 and 2.

the three network pairs using MUGs as the generative model (Figs. 6.15, 6.16, 6.17, summarized in Table 6.10). As expected the MUGs and the GC-COND models were better than INDEP. The MUGs model was also much better than GC-COND which is not surprising as the generative model obeys the assumptions of the MUGs model.

Finally, we compared the INDEP, MUGs, and GC-COND models on the three net-

	NET12			NET16			NET12-66		
	MUG	GC-COND	PVEM	MUG	GC-COND	PVEM	MUG	GC-COND	PVEM
INDEP	0/10	0/9	1/1	0/5	0/10	0/5	0/4	0/2	0/1
MUG		0/0	9/0		2/0	7/0		0/0	1/0
GC-COND			10/0			7/0			0/0

Table 6.9: Summary of structure match comparison using CONSTR generative model. The numbers p/q in each cell specify the number of times the algorithm in the row (p) beats the algorithm in the column, and the number of times the algorithm in the column beats the one in the row (q).

works using INDEP as the generative model (Figs. 6.18, 6.19, 6.20, summarized in Table 6.11). Interestingly, although INDEP was expected to do the best because the training data was from the INDEP model, the difference between the INDEP and MUGs model was marginal. In particular, INDEP beat MUGs on no more than 4 variables. This suggests that the MUGs model is able to correctly infer the condition labels. We also found that the MUGs model was again better than GC-COND.

In summary, considering all generative models and networks, the MUGs model has the most benefit. Even though INDEP beats MUGs when the generative model is INDEP, MUGs beats INDEP by a far greater margin when the generative model is MUGs. This was a somewhat surprising result given the simplicity of this model compared to GC-COND which is more flexible. This suggests that the additional flexibility in the GC-COND and PVEM models is not beneficial for these networks. Instead, the additional cost associated with learning such complex models outweighs the benefit that would be gained with the flexibility. However, we did notice that the performance margin between the different networks had a tendency to decrease as the network became more fragmented (Fig 6.8). This led to our next set of experiments where we analyze the behavior of the different models on networks of different topologies.

	NET12		NET16		NET12-66	
	MUG	GC-COND	MUG	GC-COND	MUG	GC-COND
INDEP	0/10	0/10	0/10	0/10	0/10	0/10
MUG		6/0		9/0		4/0

Table 6.10: Summary of structure match comparison using MUG generative model. The numbers p/q in each cell specify the number of times the algorithm in the row (p) beats the algorithm in the column, and the number of times the algorithm in the column beats the one in the row (q).

	NET12		NET16		NET12-66	
	MUG	GC-COND	MUG	GC-COND	MUG	GC-COND
INDEP	1/0	10/0	1/0	10/0	0/0	10/0
MUG		9/0		10/0		9/0

Table 6.11: Summary of structure match comparison using INDEP generative model. The numbers p/q in each cell specify the number of times the algorithm in the row (p) beats the algorithm in the column, and the number of times the algorithm in the column beats the one in the row (q).

6.4.2 Effect of different network topology on model performance

To test our hypothesis that the number of components in the union graph affects model performance we obtained structural and pseudo likelihood-based functional scores on a new network, NET12-79 (Fig 6.8). This network has many more connected components than the previous networks. For ease of comparison we include the scores on all four networks (Figs 6.21, 6.22, 6.23) and summarized in Table 6.12.

Using structural comparison we found that although the MUGs model was better than PVEM and GC-COND on NET16 and NET12, PVEM and GC-COND models were similar to MUG on the NET12-66 and both outperformed MUGs on NET12-79 networks. We note that the union graph of NET16 is completely connected, whereas the number of components increase in NET12-66 and NET12-79.

Using functional comparison (Figs 6.24, 6.25, 6.26), we found that although PVEM was better in general than MUGs, the number of cases on which PVEM outperformed

MUGs increased for the NET12-66 and NET12-79 networks. A similar trend was observed on comparing GC-COND against MUGs. Comparison of PVEM against GC-COND showed that GC-COND tended to have better structure match whereas PVEM tended to have functional scores. The better functional scores of the PVEM model is likely due to the generative model being more similar to PVEM than GC-COND.

The overall message from these results is that the MUGs model tends to outperform more complex models when the underlying structure of the union graph has a single giant component. However, as the number of connected components increase, models that account for more fine-grained granularity of condition-specificity do better.

Algorithm 7 Search procedure for GC-COND

```
Input:
  Random variable set, $\mathbf{X} = \{X_1, \cdots, X_{|\mathbf{X}|}\}$
  Dataset of RV joint assignments, $\mathbf{D} = \{D_1, \cdots, D_K\}$
Output:
  Inferred graphs $\mathcal{G} = \{G_1, \cdots, G_k\}$
Initialize $CC_{\widehat{G}}$ to size $1$ graph components, one component per variable
while Score not converged do {/*Modify graph structure*/}
  for $X_i \in \mathbf{X}$ do
    for $X_j \in \mathbf{X} \setminus \{X_i\}$ do
      for $l \in \mathcal{C}$ do
        if $X_i$ and $X_j$ are not in the same component then
          Estimate new $\gamma$'s as the weighted sum of the $\gamma$'s of the graph component of
          $X_i$ and $X_j$.
        end if
        Estimate the score improvement using Eq 6.29
      end for
    end for
    $\{X_j, l\} = \arg\max_{X'_j \neq X_i, l'} \Delta S_{\{X_i, X'_j\}, l'}$
  end for
  for $X_i \in \mathbf{X}$ do
    Find best new pair $\{X_j, l\}$, $1 \leq l \leq K$ that maximizes $\Delta S_{ij}$ using Eq 6.19
  end for
  Update graphs and $CC_{\widehat{G}}$ by adding edges in decreasing order of score improvement.
  while Score not converged do {/*EM on parameters*/}
    Expectation step to re-estimate $\gamma_{kcd}, 1 \leq k \leq K, 1 \leq c \leq |CC_{\widehat{G}}|, 1 \leq d \leq |\mathbf{D}|$
    Maximization step to re-estimate $\alpha_{kc}$ and parameters of the conditionals
    $P(X_i|\mathbf{M}_{ki})$ using the new $\gamma_{kcd}$'s
  end while
end while
```

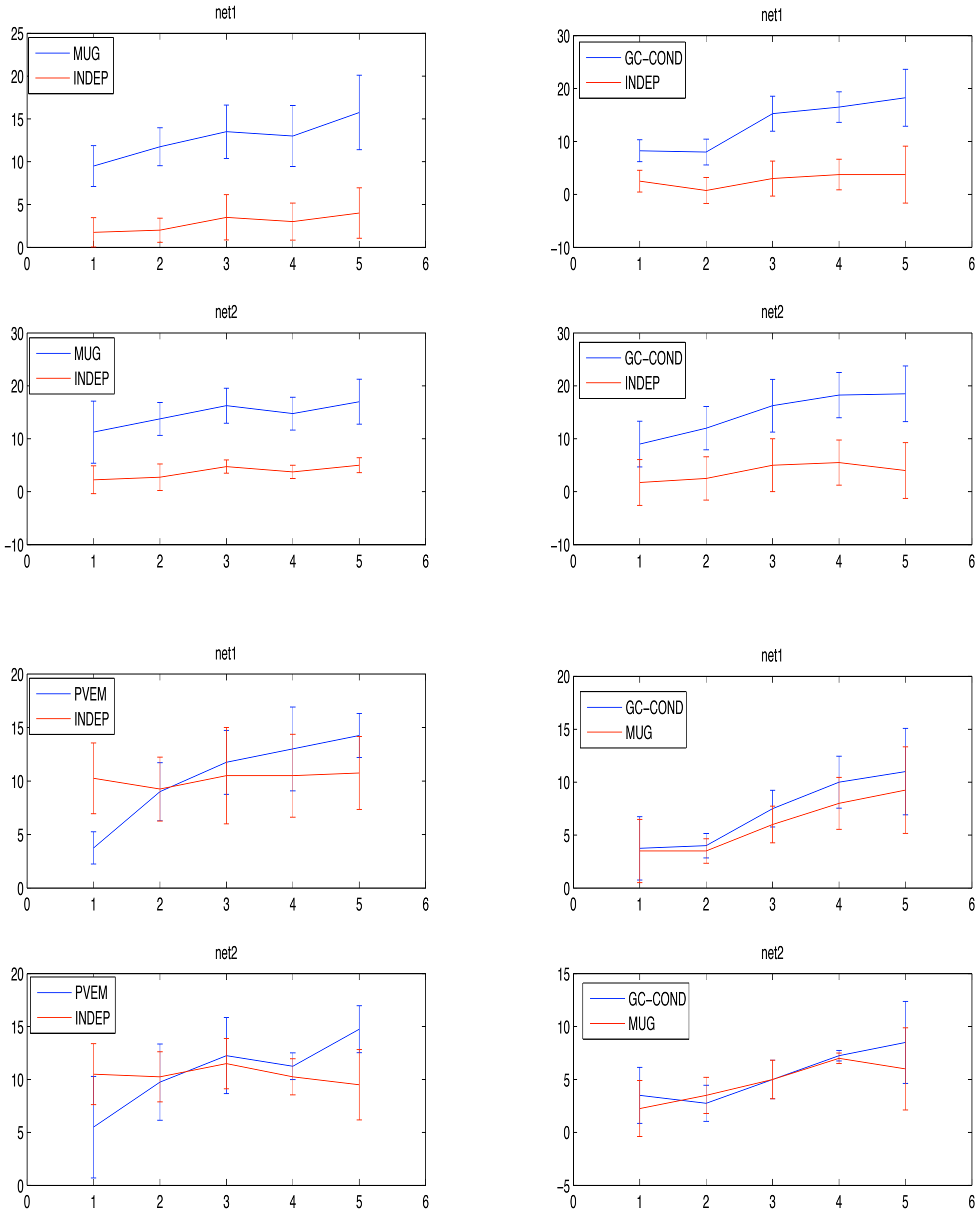


Figure 6.9: Comparison of different models on network pair NET12. Generative model is CONSTR and comparisons are based on structural match to the true network. x-axis is the increasing folds of the training data. The training data set size decreases with increasing fold count. y-axis is the number of variables on which one method is significantly better than the other. Continued on next page

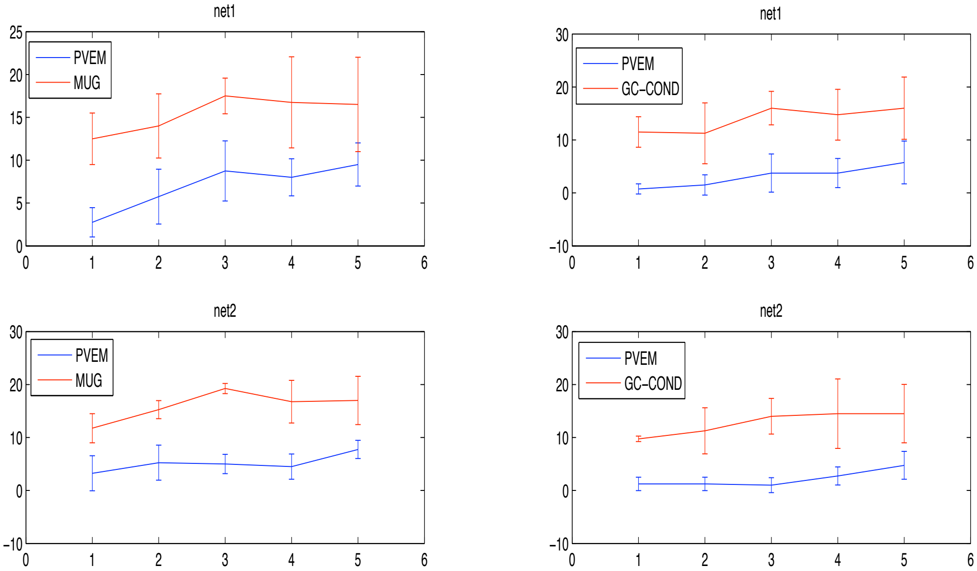


Figure 6.10: Comparison of different models on network pair NET12 contd. Generative model is CONSTR and comparisons are based on structural match to the true network. x-axis is the increasing folds of the training data. The training data set size decreases with increasing fold count. y-axis is the number of variables on which one method is significantly better than the other.

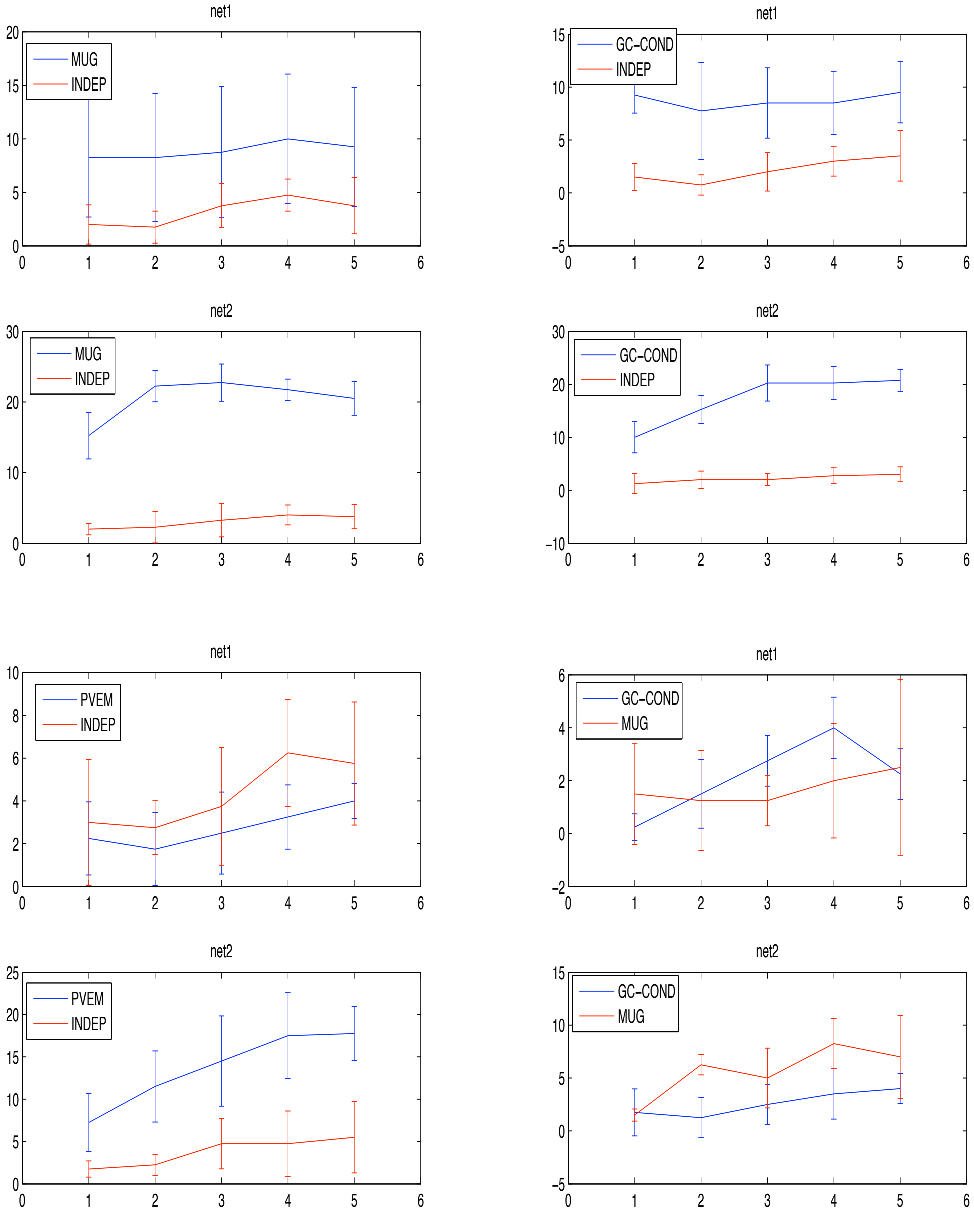


Figure 6.11: Comparison of different models on network pair NET16. Generative model is CONSTR. The remaining legend is same as 6.9. Continued on next page.

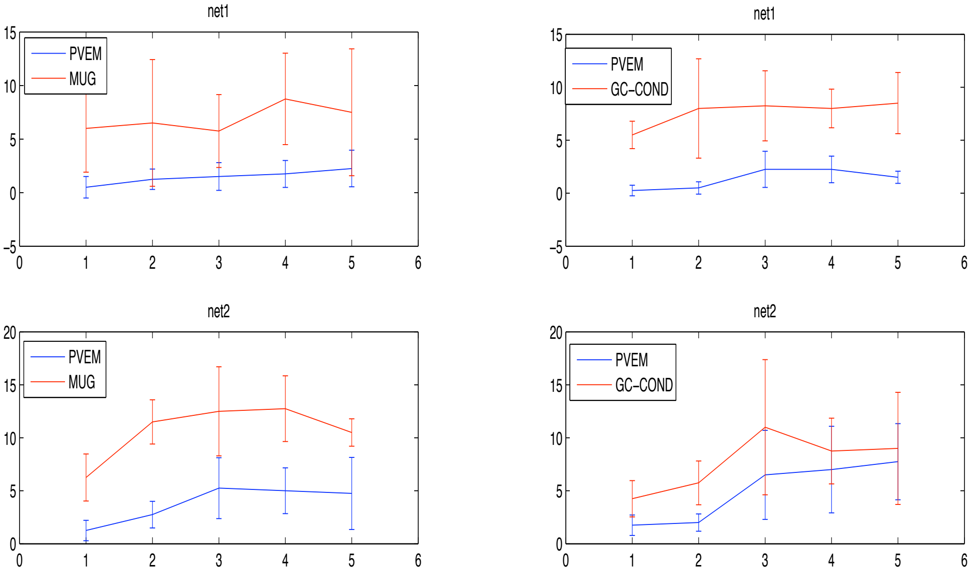


Figure 6.12: Comparison of different models on network pair NET16 contd. Generative model is CONSTR. The remaining legend is same as 6.9.

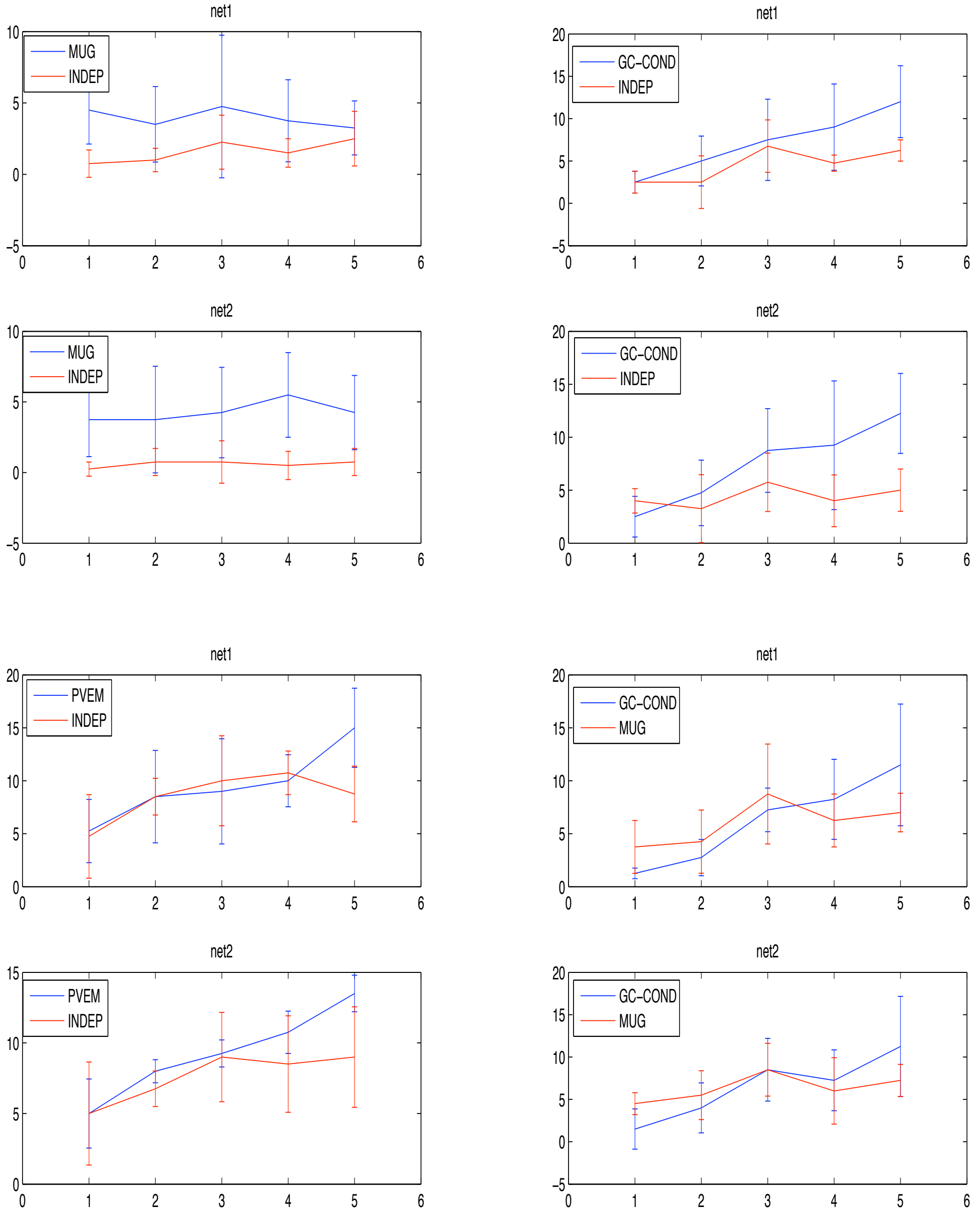


Figure 6.13: Comparison of different models on network pair NET12-66. Generative model is CONSTR. The remaining legend is same as 6.9. Continued on next page.

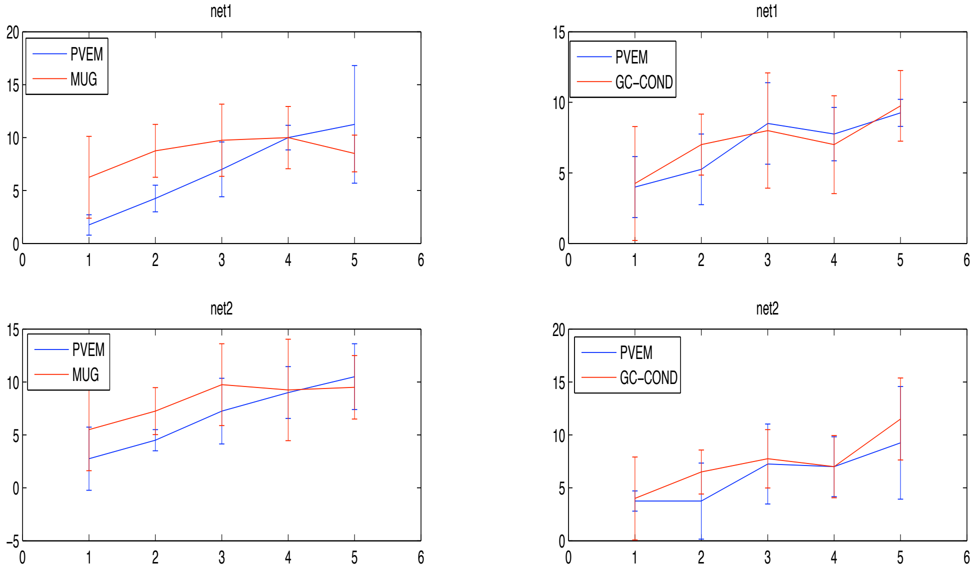


Figure 6.14: Comparison of different models on network pair NET12-66 contd. Generative model is CONSTR. The remaining legend is same as 6.9.

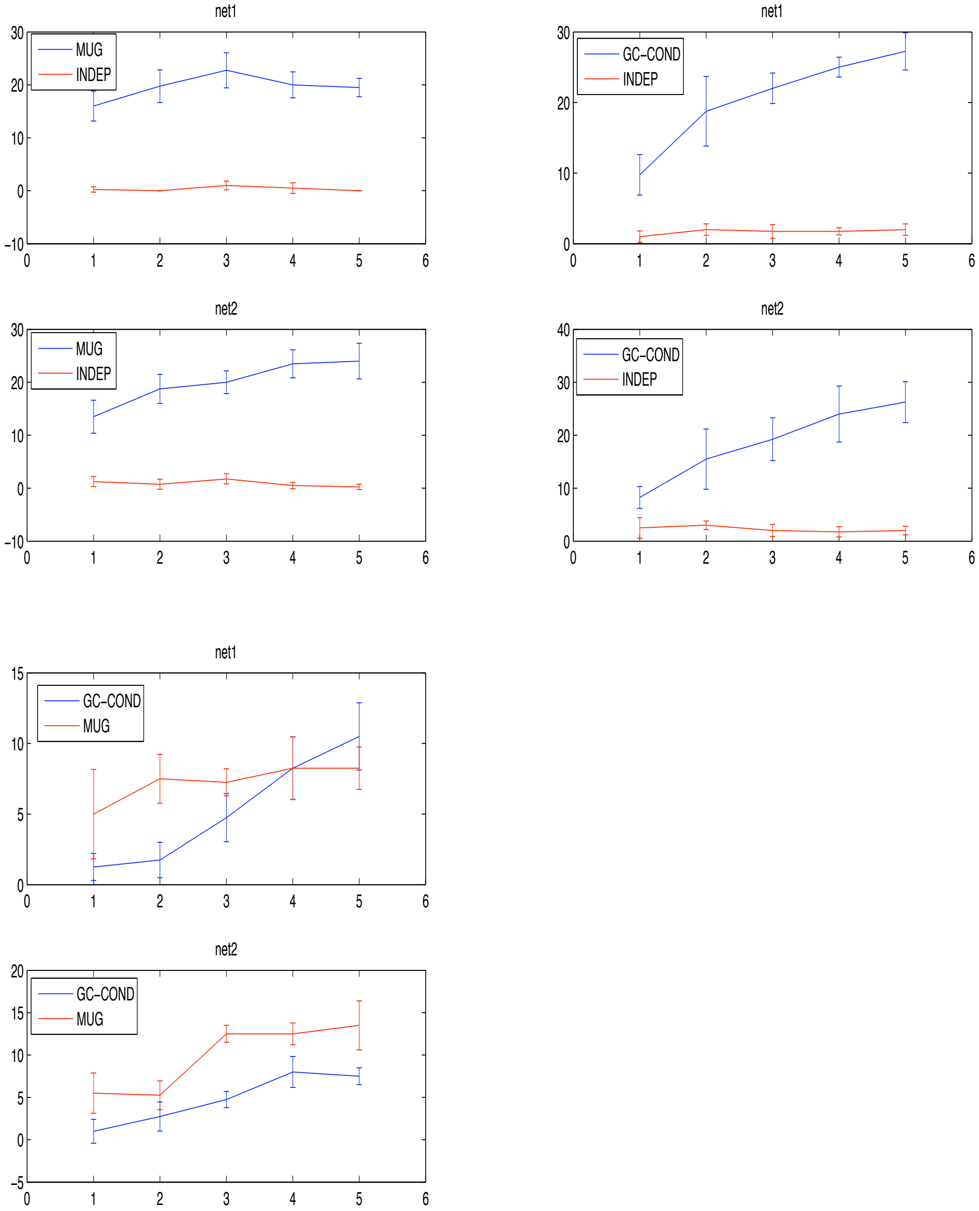


Figure 6.15: Structural comparison of different models on network pair NET12 using the MUGs generative model. The remaining legend is same as 6.9.

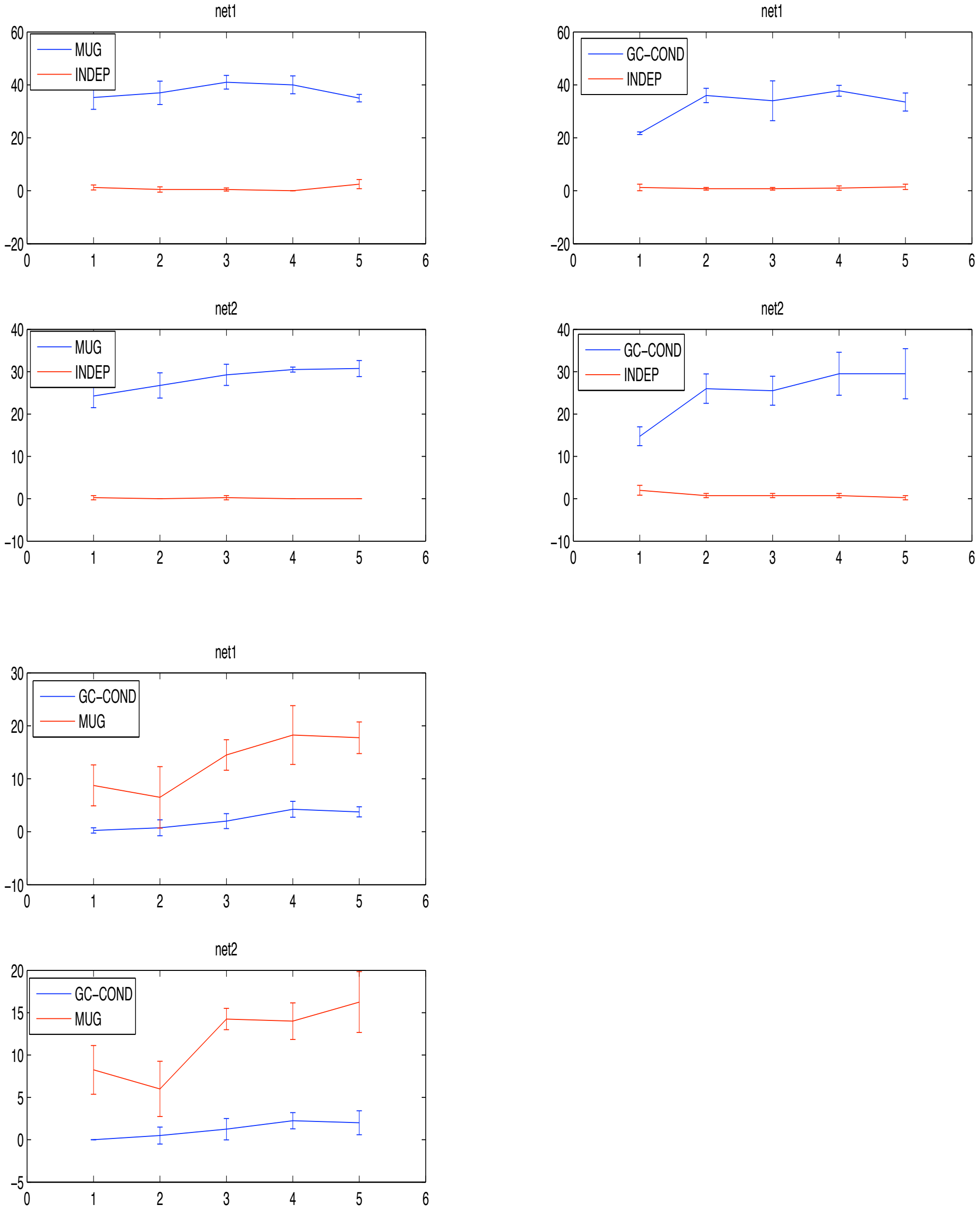


Figure 6.16: Structural comparison of different models on network pair NET16 using the MUGs generative model. The remaining legend is same as 6.9.

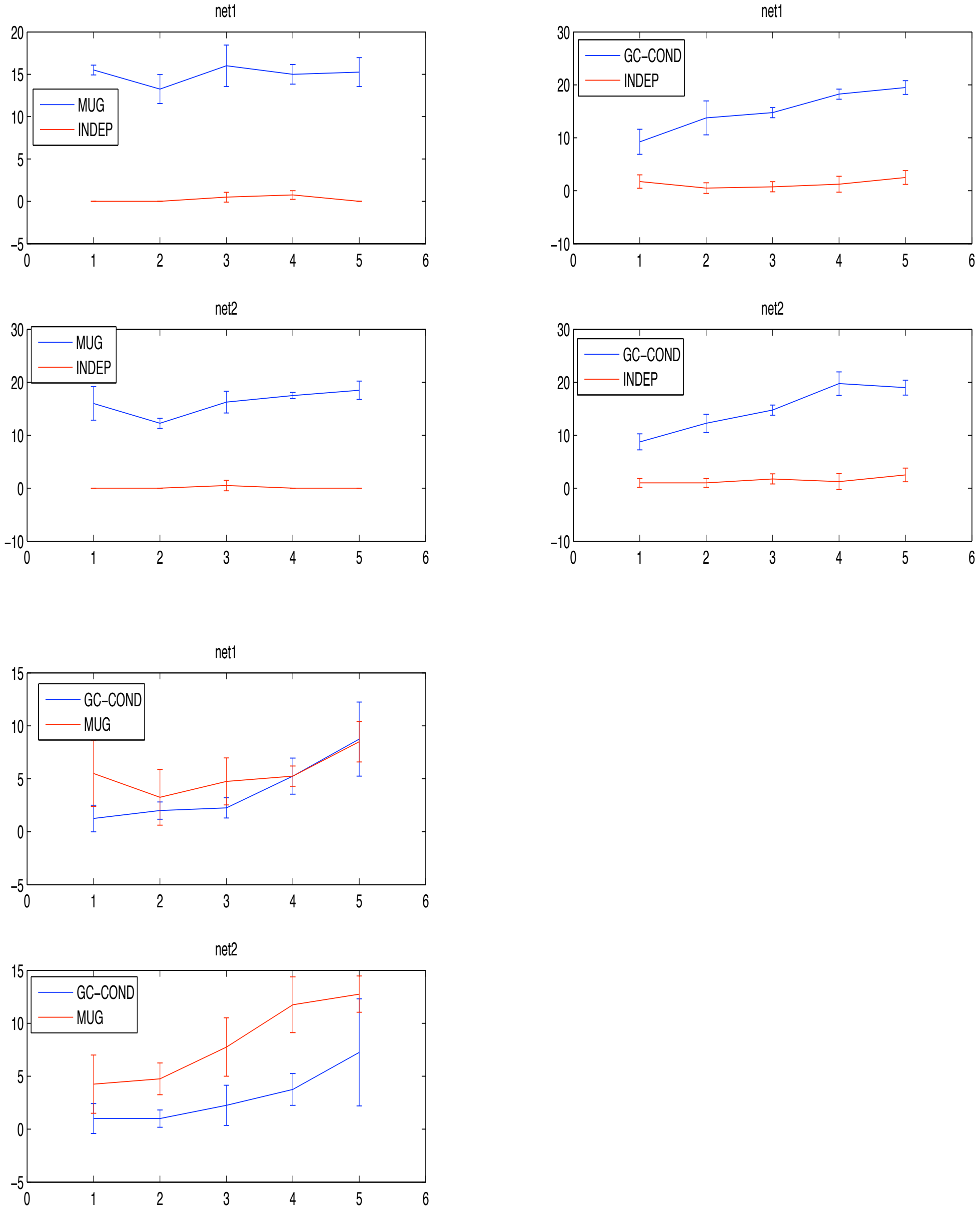


Figure 6.17: Structural comparison of different models on network pair NET12-66 using the MUGs generative model. The remaining legend is same as 6.9.

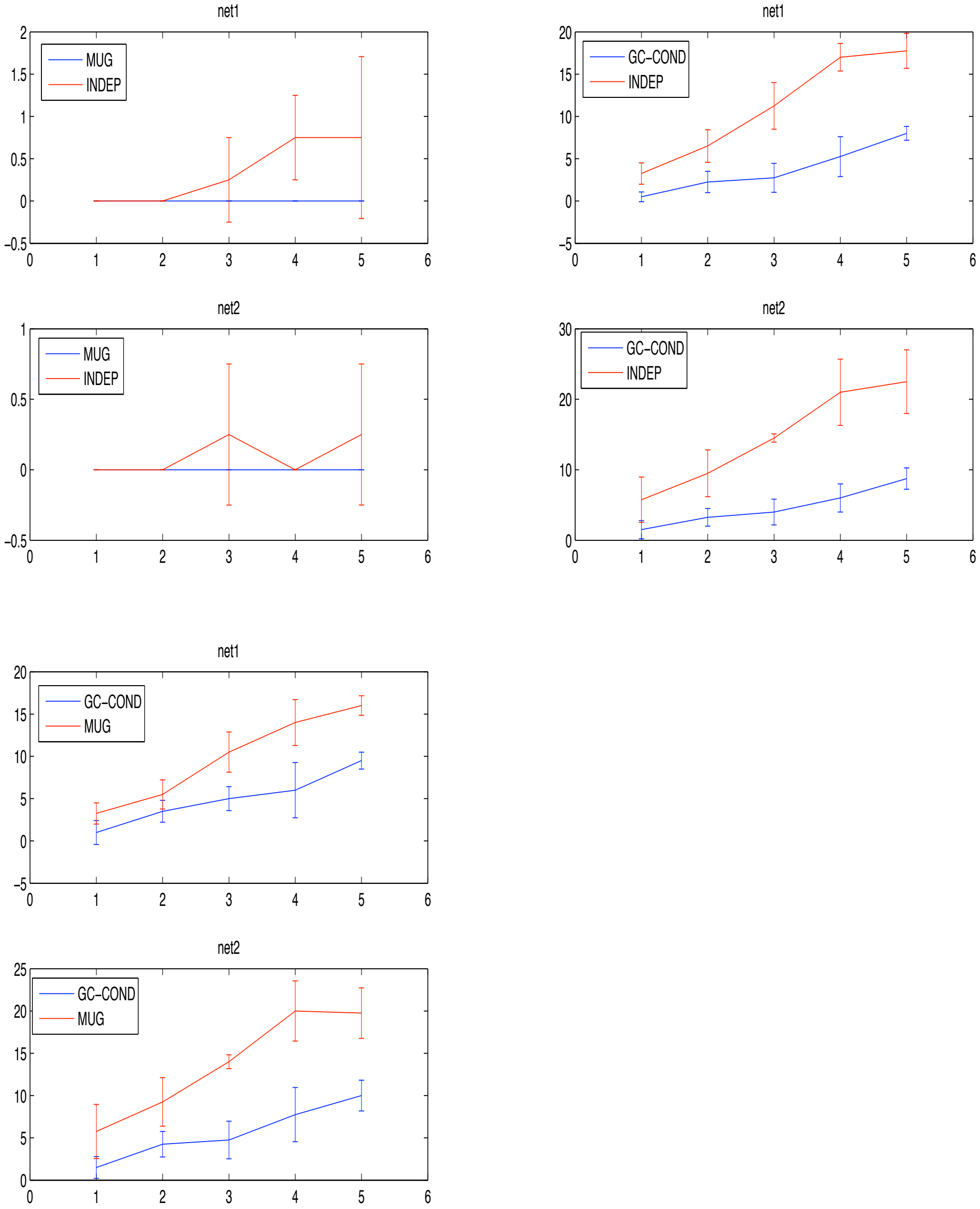


Figure 6.18: Structural comparison of different models on network pair NET12 using the INDEP generative model. The remaining legend is same as 6.9.

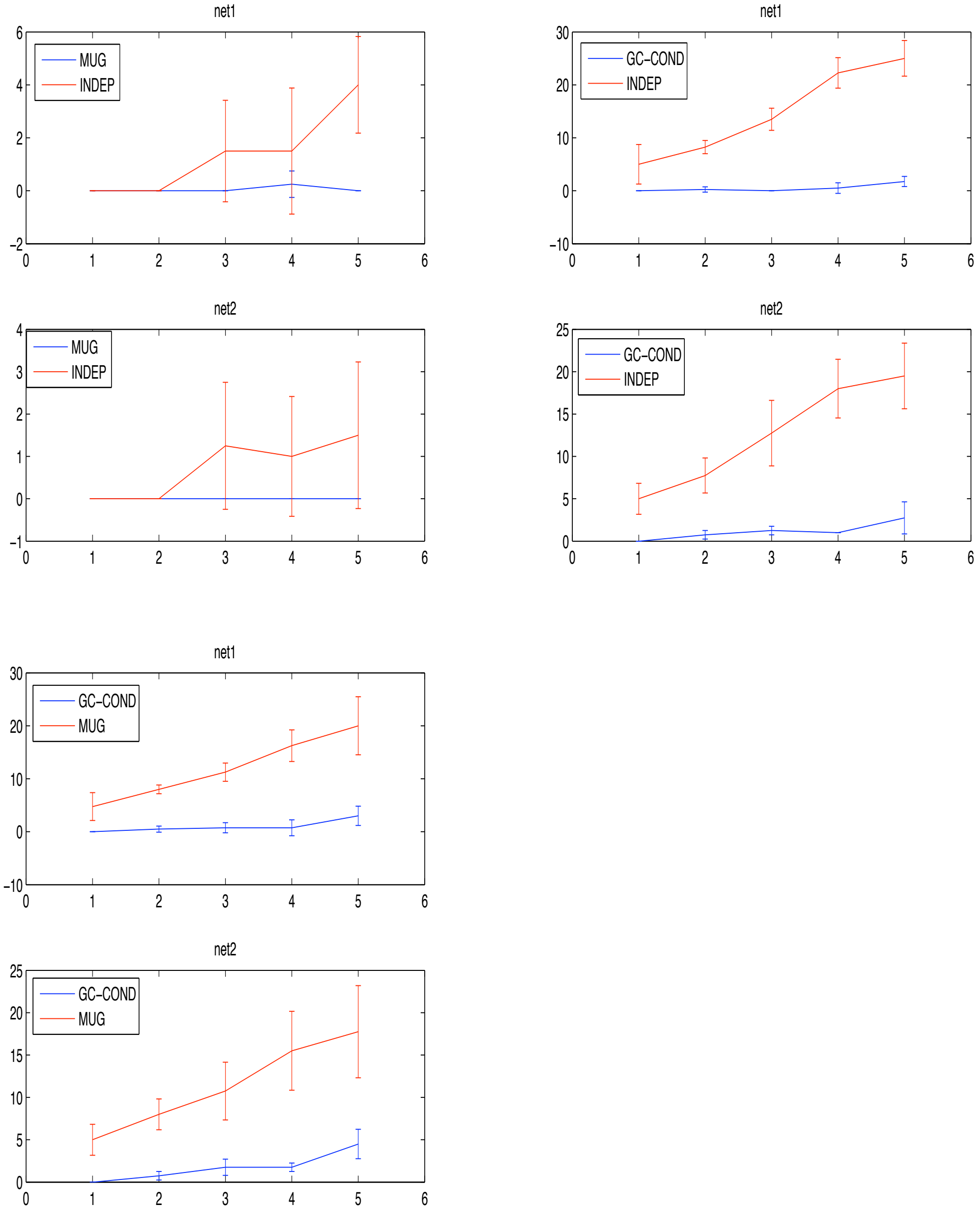


Figure 6.19: Structural comparison of different models on network pair NET16 using the INDEP generative model. The remaining legend is same as 6.9.

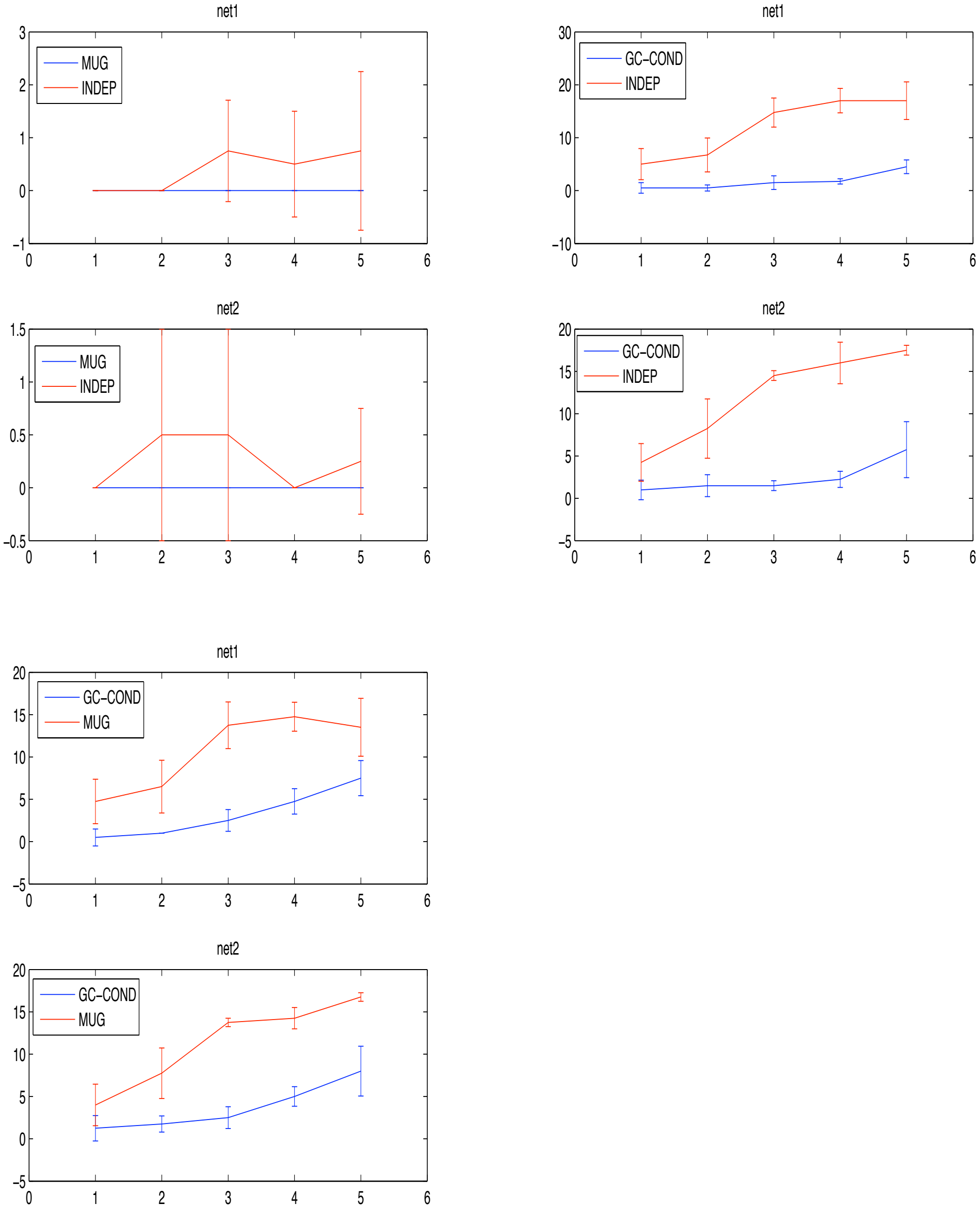
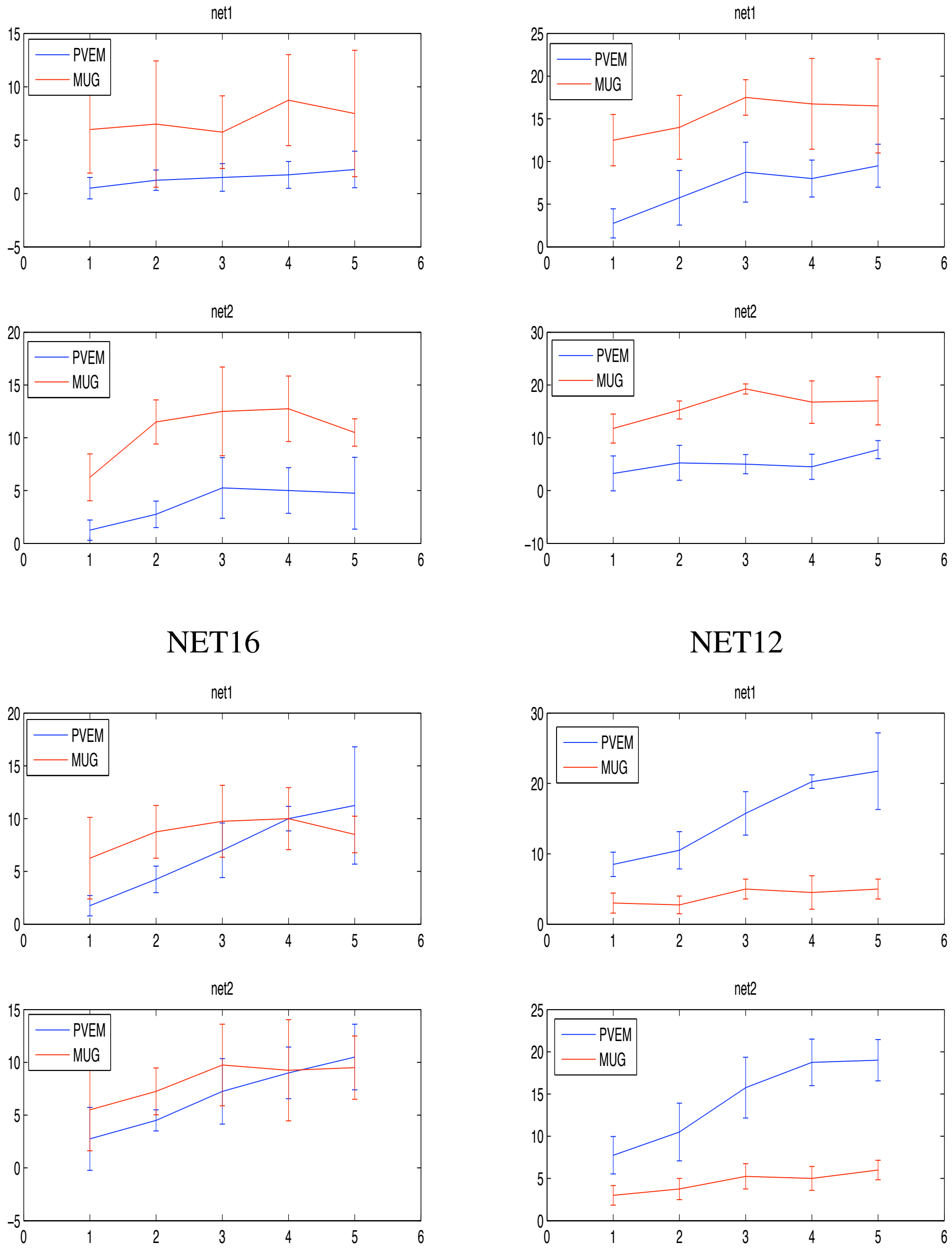


Figure 6.20: Structural comparison of different models on network pair NET12-66 using the INDEP generative model. The remaining legend is same as 6.9.



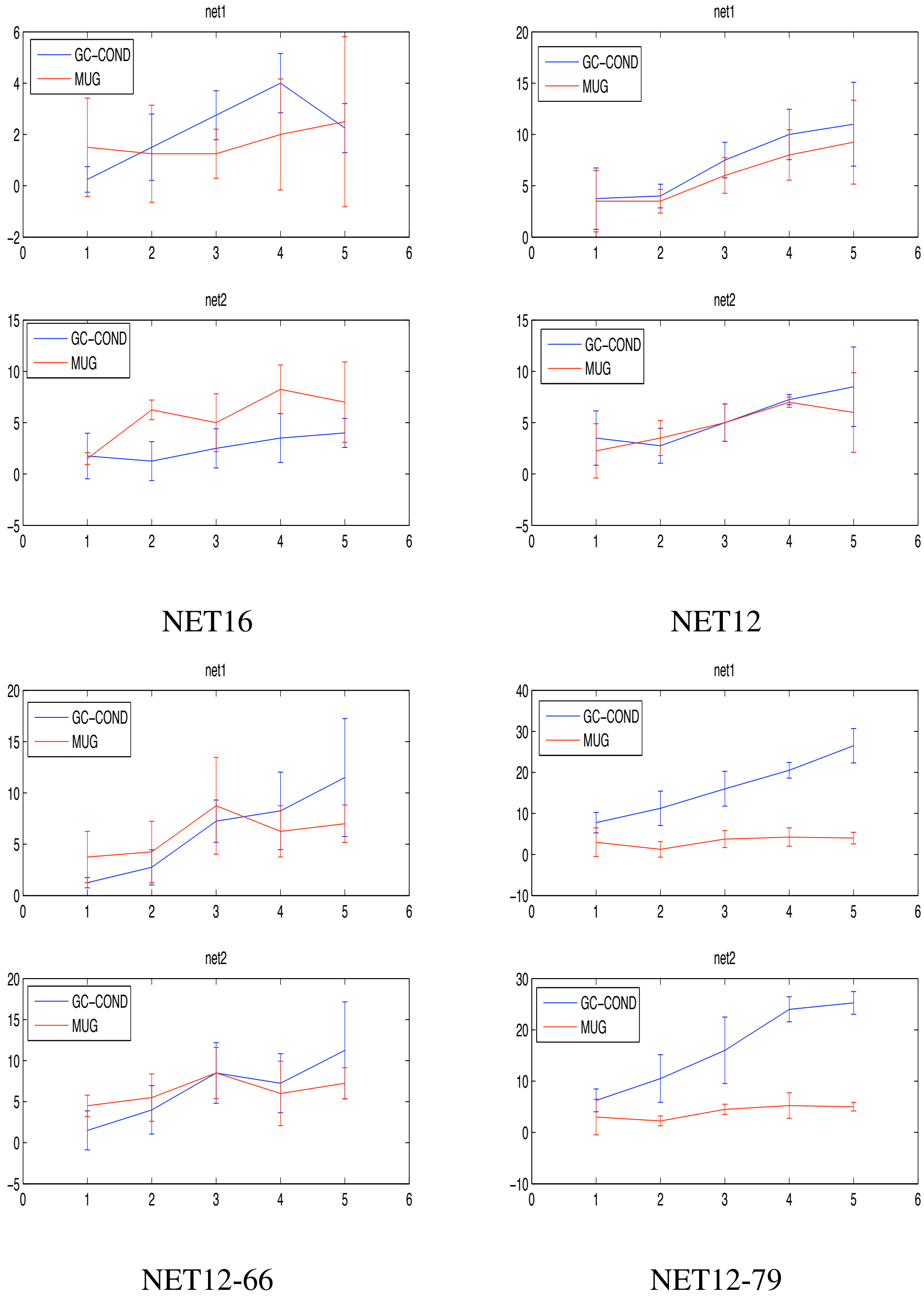
NET16

NET12

NET12-66

NET12-79

Figure 6.21: Structural comparison of MUGs vs PVEM on networks with different topologies. x-axis is the increasing folds of the training data. The training data set size decreases with increasing fold count. y-axis is the number of variables on which one method is significantly better than the other.



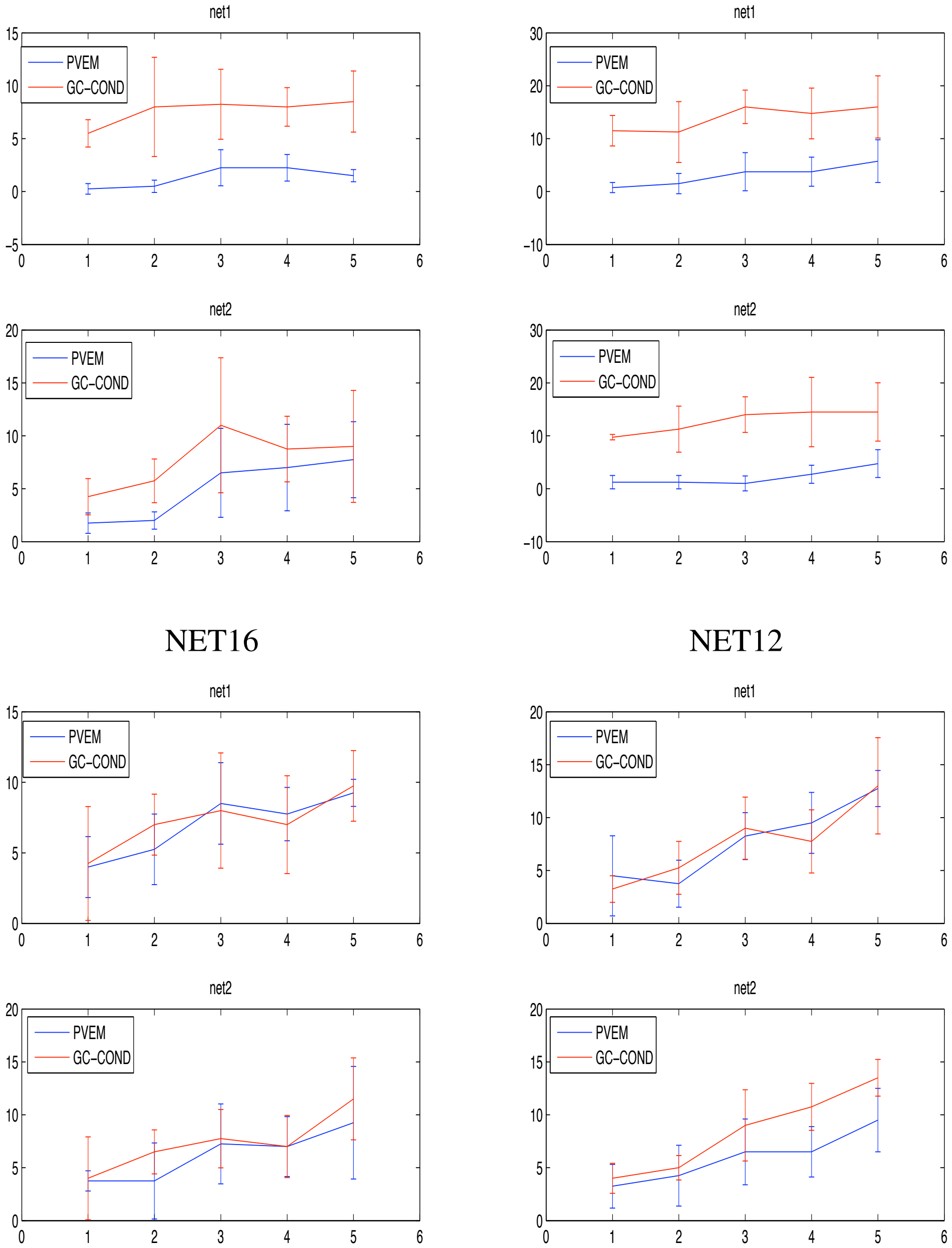
NET16

NET12

NET12-66

NET12-79

Figure 6.22: Structural comparison of GC-COND vs MUG on networks with different topologies. x-axis is the increasing folds of the training data. The training data set size decreases with increasing fold count. y-axis is the number of variables on which one method is significantly better than the other.



NET16

NET12

NET12-66

NET12-79

Figure 6.23: Structural comparison of PVEM vs GC-COND on networks with different topologies. x-axis is the increasing folds of the training data. The training data set size decreases with increasing fold count. y-axis is the number of variables on which one method is significantly better than the other.

	Structure match				Functional match			
	NET16	NET12	NET12-66	NET12-79	NET16	NET12	NET12-66	NET12-79
MUG/PVEM	7/0	9/0	1/0	0/10	0/4	0/2	0/10	0/10
MUG/GC-CO	2/0	0/0	0/0	0/8	0/0	0/3	0/10	0/9
PVEM/GC-CO	0/7	0/10	0/0	0/0	4/0	6/0	0/0	8/0

Table 6.12: Summary of structure and function comparison of the MUG, GC-COND and PVEM models on networks of different topology. The numbers p/q in each cell specifies the number of times algorithm before the '/' is better than the algorithm after the '/'.

6.4.3 Parameter tying

We compared the graph component models with (GC-CONDSH) and without (GC-COND) parameter tying on four network pairs with different amounts of shared edges (Table 6.8). The two models were compared based on overall structure and pseudo likelihood-based functional match (Figs 6.27 and 6.28). We did not find parameter tying to have significant advantage over the model without parameter tying.

Next, we compared these models based the number of correctly captured shared edges as a function of decreasing amounts of training data (Fig 6.29). We found that although GC-COND had a tendency to outperform GC-CONDSH for NET12, both models were at par on NET13 and GC-CONDSH was better than GC-COND on NET16. Although NET14 appeared to not obey this trend, on the whole this suggest that as we decrease the number of shared edges in the networks, GC-CONDSH does a better job of getting the shared edges right. The task of finding the shared edges is harder for NET16 than it is for NET12. The fact that GC-CONDSH does a better job of getting the shared edges when the amount of similarity between networks decreases indicates that parameter tying is beneficial for learning shared structure

Finally, we examined the stability of the inferred networks by the two methods (Figs. 6.30 and 6.31). Here we obtained the average F-score between all pairs of networks learned using different folds of the data. For example for $t = 4$ folds, there are six pairs of network 1 and six pairs of network 2. We computed the F-score for these pairs and took their average. A higher F-score suggests more stability of the inferred structures. We performed this procedure first considering all edges (Fig 6.30), and then considering only edges that matched with the true networks (Fig 6.31). We found that GC-CONDSH had a tendency to get more stable structures, especially when considering only the matched edges. This suggests that parameter tying may also be useful for learning more stable structures.

In summary, we found that although parameter tying did not give us overall better structures, it does help to learn shared edges better, especially when the amount of similarity between the involved networks is small. Parameter tying also enables us to learn more stable structures.

6.5 Discussion

In this chapter we have described and evaluated different methods for learning condition-specific networks. All these models learn networks providing fine-grained interaction information among the random variables (genes) underlying condition-specific response. These models extend the INDEP learning model in different directions.

The various NIPD approaches that assume the condition variable for each dataset is known, extends existing approaches by allowing us to infer networks for any subset of conditions. For example, if we have two conditions A and B, existing approaches either find only the patterns that are exclusive or specific to A and specific to B, or find only the patterns that are both common to A and B. In contrast, we are able to simultaneously find the patterns both unique and common to different conditions. When we consider learning in subsets of conditions, we automatically pool the data from these conditions. We demonstrated using simulated data that the NIPD approaches are able to learn better quality structures and also learn the shared structure much better especially in the face of limited training data. These results highlight the benefits of exploiting shared information by data pooling during structure learning.

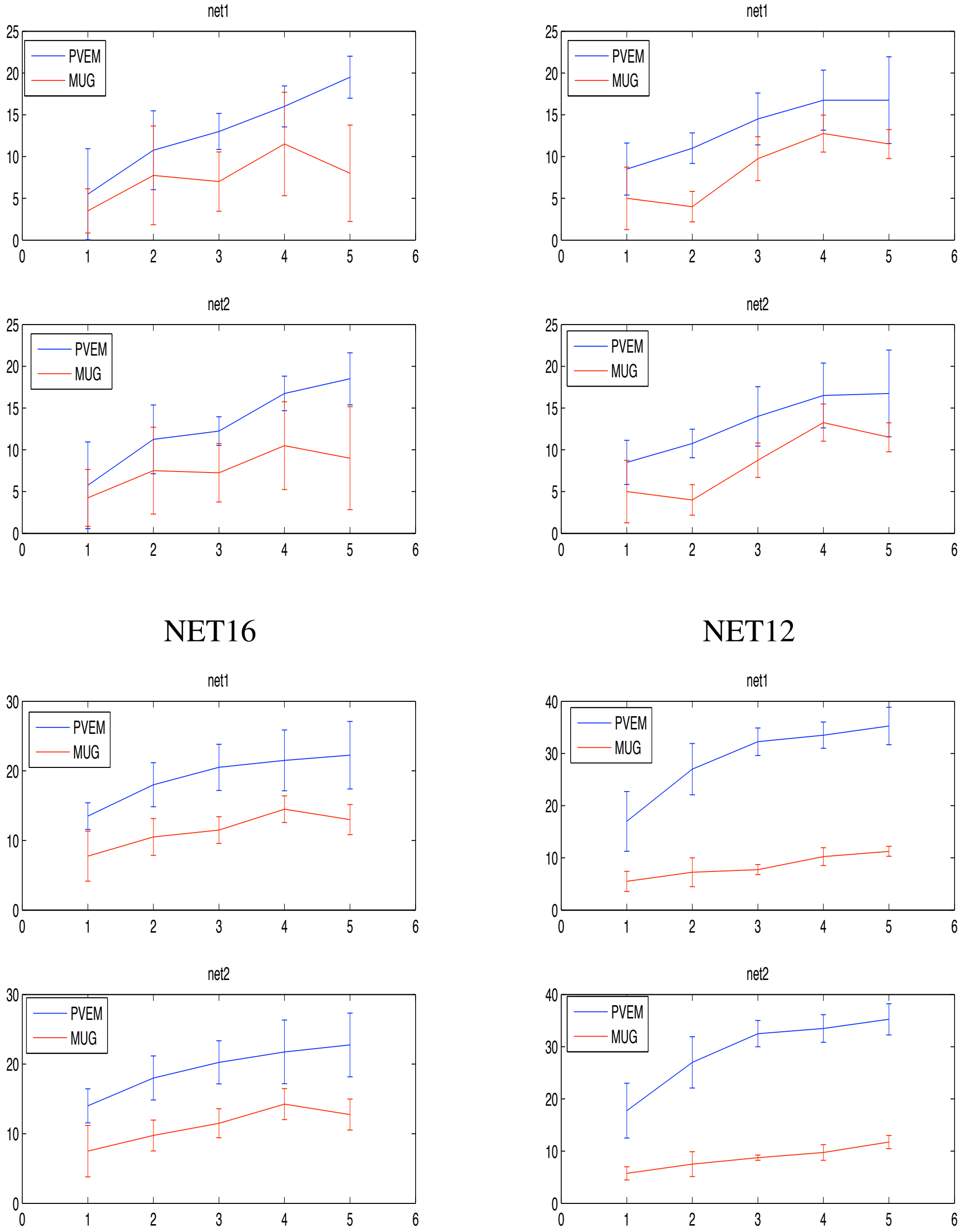
The important modeling question that arises when working with these models is to be able to describe the conditional distributions of each of the random variables that exploits the subset information. We derived two main variations to the conditionals and found that both forms were better at capturing shared structure. However, the model that uses a product of the different condition-set-specific Markov blankets had the best performance. This was the simplest model and the reason of it success is likely to be its ability to capture weakly shared edges that are identified only when we consider a condition set alone, but not when we include the subsets within it as done by the weighted sum model.

We then considered models that extend the INDEP model by having the flexibility of automatically inferring the condition variable for each data point. We consider several variations of these hidden condition variable models, where the condition variable is allowed to influence different parts of the network. The MUGs model is the simplest, with a single condition variable for the entire graph. The PVEM model is the most complex, with a condition variable for each random variable. Finally, the GC-COND model is intermediate between the MUG and PVEM models and has a condition variable for each connected component of the union graph. Comparative analysis of these models on simulated data yielded some surprises. In particular, we found that the MUG model performed better than the GC-COND and PVEM models, even when the data was generated from the CONSTR generative model, which is more similar to PVEM (and GC-COND) than it is to MUG. MUG was also close to INDEP when the generative model was the same as INDEP. This made MUG, the simplest model, to be the overall winner when compared on the NET12 and NET16 datasets. However the performance difference was small when on the NET12-66 dataset. One reason for this behavior is that both NET12 and NET16 had the majority of nodes connected in a single giant component within the union graph, whereas NET12-66 had two medium sized components. This led us to ask if the topology of the union graph influences the model performance.

If the union graph has only one or a single giant component, the condition assignment to all the nodes in the networks from the two conditions gets constrained. In particular, if a node X_1 is connected to X_2 in condition A and to X_3 in condition B, if X_1's condition is assigned to A, then for the $\{X_1, X_2\}$ dependency to hold in condition A, X_2 must also be assigned condition A. Similarly, while considering the condition assignment to X_3, because the dependency $\{X_3, X_1\}$ must hold only in condition B, and X_1 is already assigned to A, for the dependency to not hold, X_3 must be assigned condition B. If X_3 is assigned B while X_1 is assigned A, while drawing a sample for X_3 we will use X_1's value from condition A, but in reality X_1's value should be from condition B. In general this suggests that the condition variable of the entire connected component of the union graph is constrained. This constraint is automatically imposed in the MUGs model if the true union graph had a single connected component. In contrast, the PVEM and GC-COND models must infer these and impose the constraints. PVEM does not impose any constraints, and therefore suffers more than GC-COND. But both models are more complex than MUGs, and the benefits of model complexity are not as much as the difficulty in learning a constrained network required for the NET12 and NET16 datasets. In contrast the constraints imposed by the union graphs in both NET12 and NET16 are easily satisfied by the MUGs model which infers a single condition variable anyway forcing all nodes in the network to be assigned the same condition.

We tested the hypothesis that model performance depends upon the topology of the union graph by using a new network pair, NET12-79 which had many connected components. We found that both GC-COND and PVEM models had better performance on this network. This suggests that the conditions where the union graph is fragmented are likely to exhibit complex condition-specific behavior with different subsets of genes having different condition-specific behavior. These are the types of cases where the GC-COND and PVEM models are likely to be beneficial.

We finally assessed the benefit of parameter tying in the GC-COND model. The overall structures and associated functional scores were not significantly different in models with and without parameter tying. However, when the amount of similarity between networks of the two conditions decreased, GC-CONDSH, which implements parameter tying, was better at learning shared edges than GC-COND, which does not perform any parameter tying. In datasets with low proportion of shared edges among the network structures, capturing the shared part is harder than in datasets with high proportion of shared edges. Being able to capture shared edges in network pairs with low similarity suggests that parameter tying helps to capture shared edges when it is harder to do so. We also found that the GC-CONDSH to produce more stable structures. However, the GC-CONDSH models are more complex than GC-COND. Our results were only modestly encouraging for the GC-CONDSH model. It may be that the learning problems in our simulations were simple enough to be learned without parameter tying, and the benefit fine-tuned data pooling done by parameter tying in the GC-CONDSH is not necessary. Instead this imposes additional overhead of learning the parameters in a tied situation (split weight vectors), and may lead to less accurate parameters.



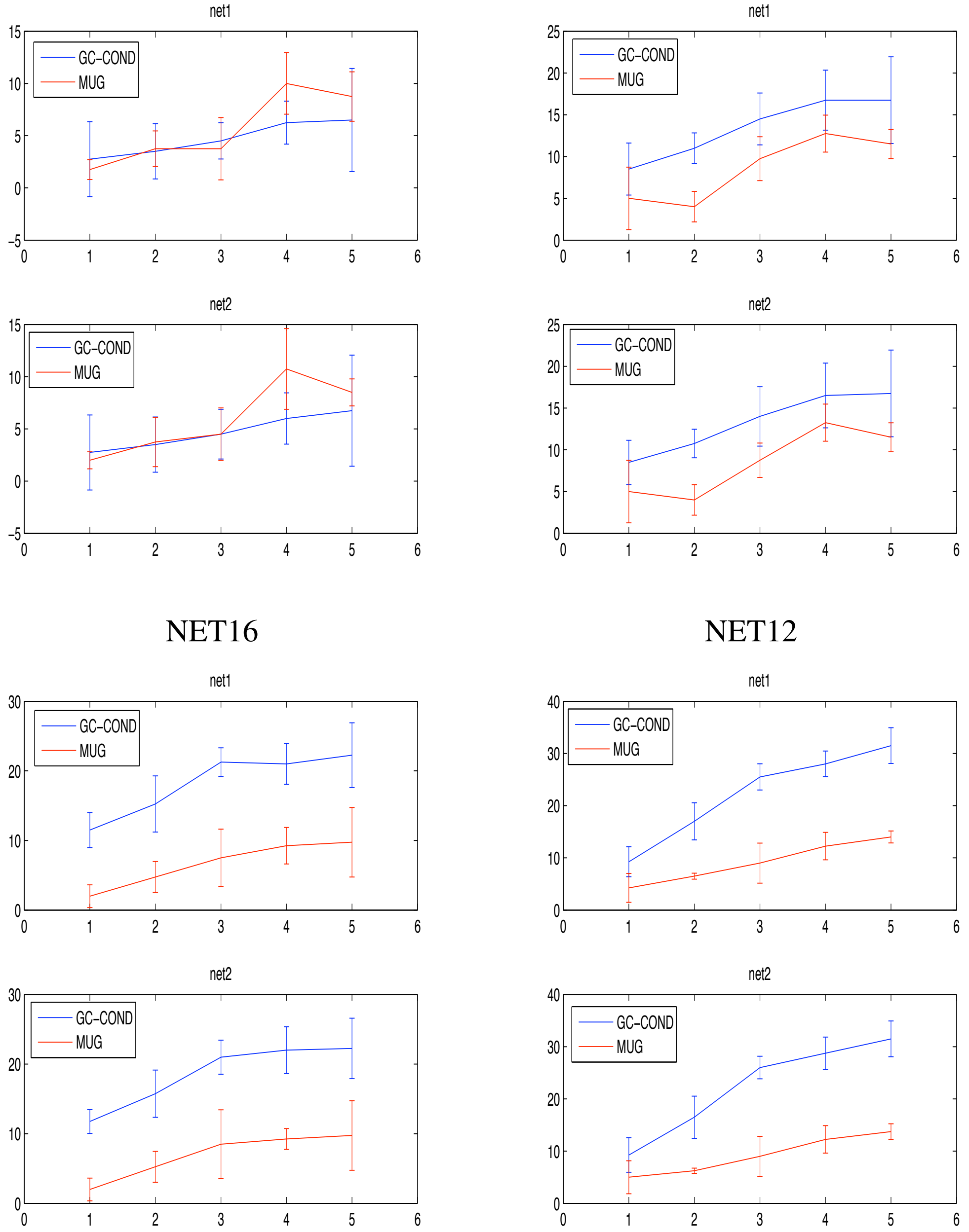
NET16

NET12

NET12-66

NET12-79

Figure 6.24: Functional comparison of PVEM vs MUG models on networks with different topologies. Generative model is CONSTR and comparisons are based on pseudo likelihood match to the true distribution. The x-axis corresponds to the number of folds of the training data. The training data size decreases as we increase the number of folds. y-axis is the number of variables on which one algorithm has a significantly higher pseudo likelihood score than the other.



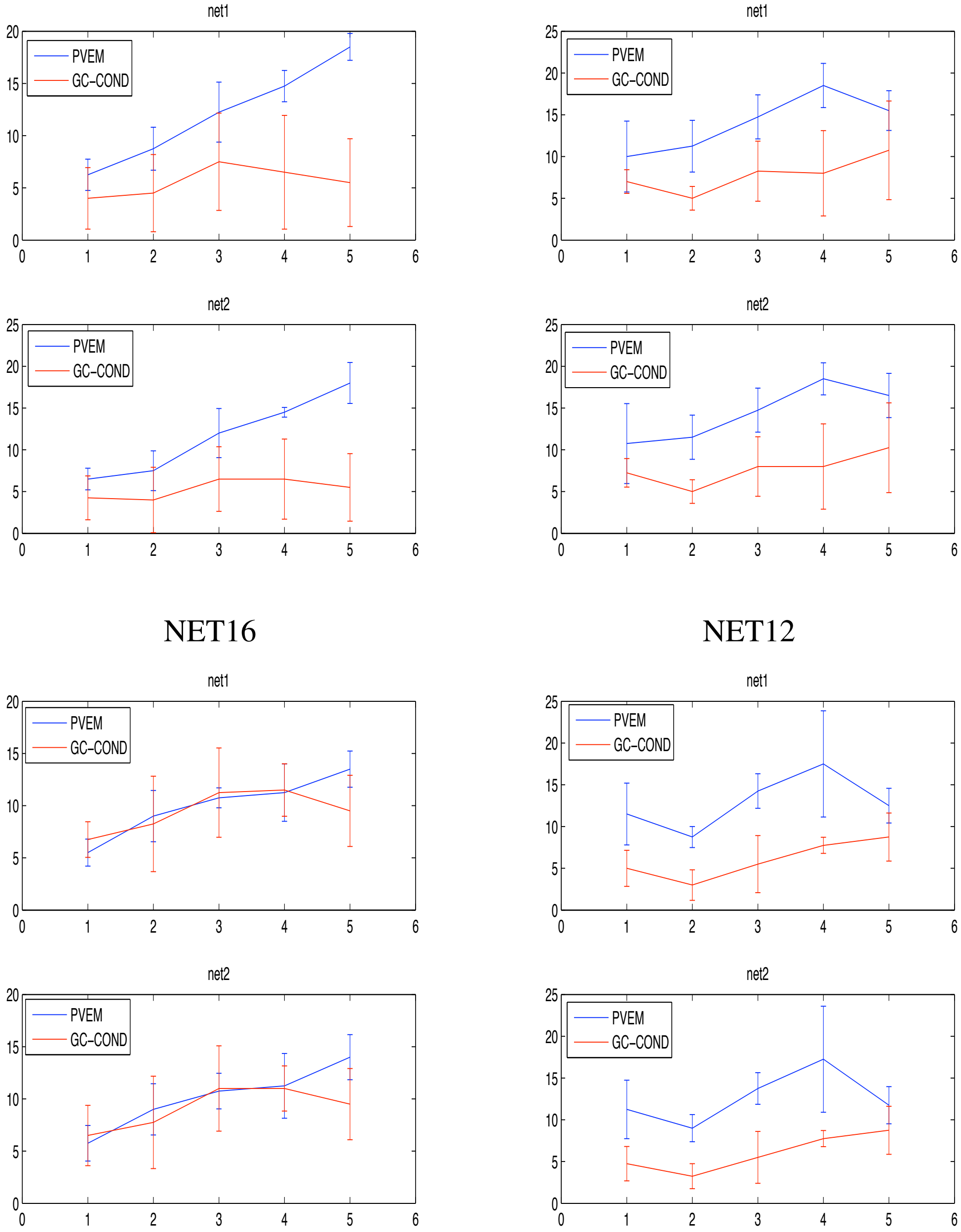
NET16

NET12

NET12-66

NET12-79

Figure 6.25: Functional comparison of GC-COND vs MUG model on networks with different topologies. Generative model is CONSTR and comparisons are based on pseudo likelihood match to the true distribution. The x-axis corresponds to the number of folds of the training data. The training data size decreases as we increase the number of folds. y-axis is the number of variables on which one algorithm has a significantly higher pseudo likelihood score than the other.



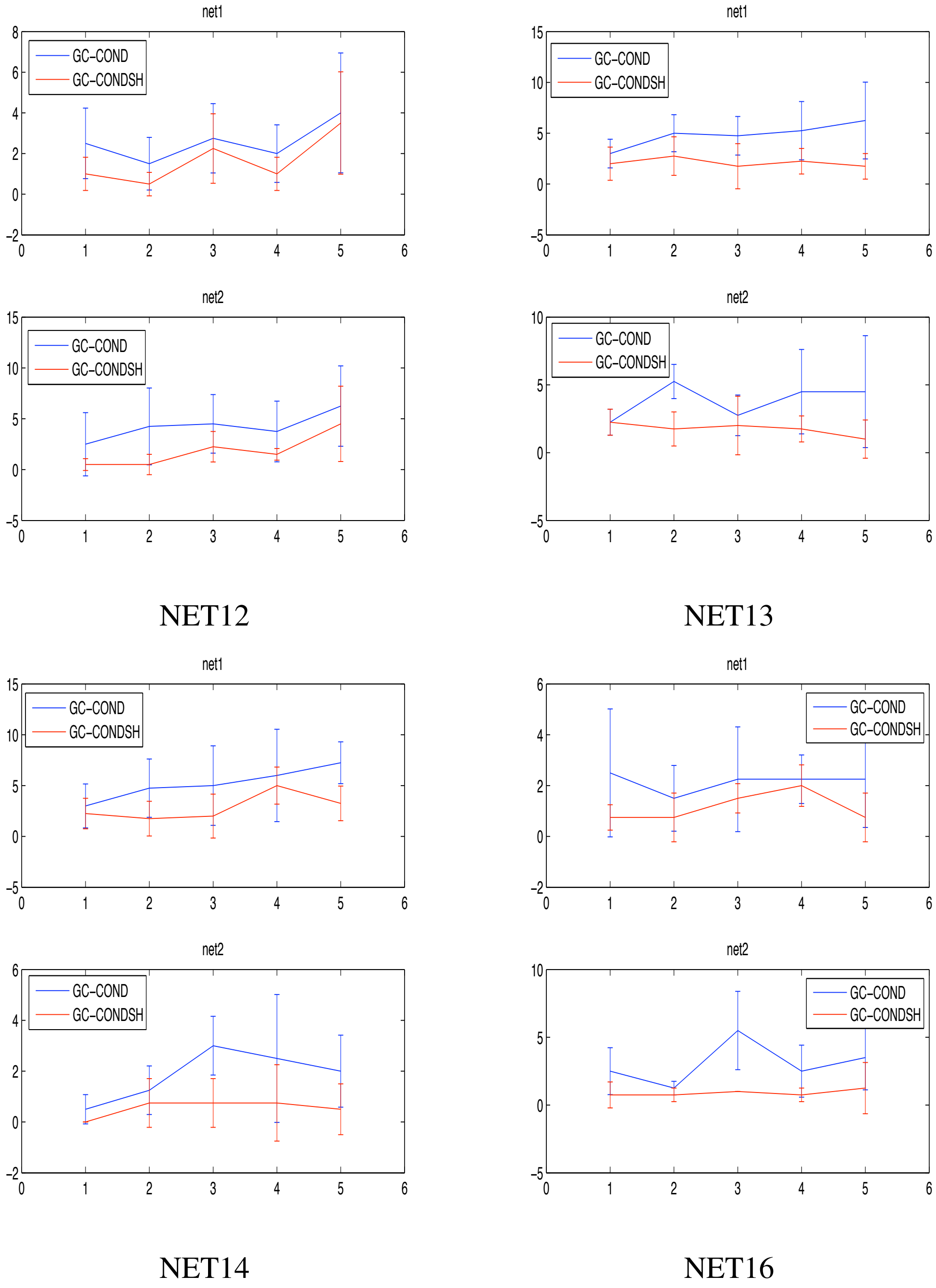
NET16

NET12

NET12-66

NET12-79

Figure 6.26: Functional comparison of PVEM vs GC-COND models on networks with different topologies. Generative model is CONSTR and comparisons are based on pseudo likelihood match to the true distribution. The x-axis corresponds to the number of folds of the training data. The training data size decreases as we increase the number of folds. y-axis is the number of variables on which one algorithm has a significantly higher pseudo likelihood score than the other.



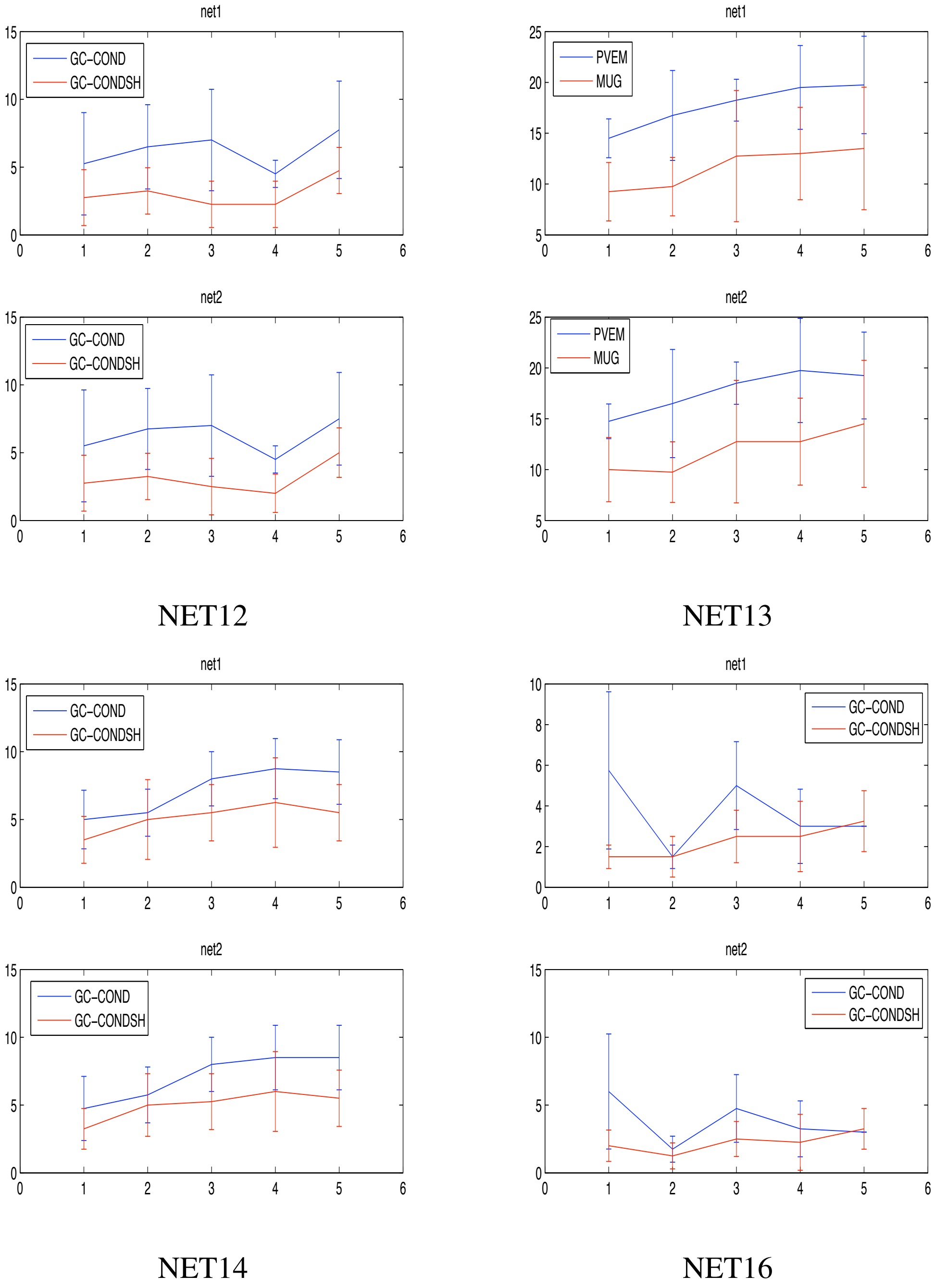
NET12

NET13

NET14

NET16

Figure 6.27: Structural comparison of models with (GC-CONDSH) and without parameter tying (GC-COND) on network pairs with varying similarity (NET12, NET13, NET14, NET16). x-axis corresponds to training data folds and y-axis corresponds to number of variables one method is significantly better than another.



NET12

NET13

NET14

NET16

Figure 6.28: Pseudo likelihood based functional comparison of models with (GC-CONDSH) and without parameter tying (GC-COND) on network pairs with varying similarity (NET12, NET13, NET14, NET16). x-axis corresponds to training data folds and y-axis corresponds to number of variables one method is significantly better than another.

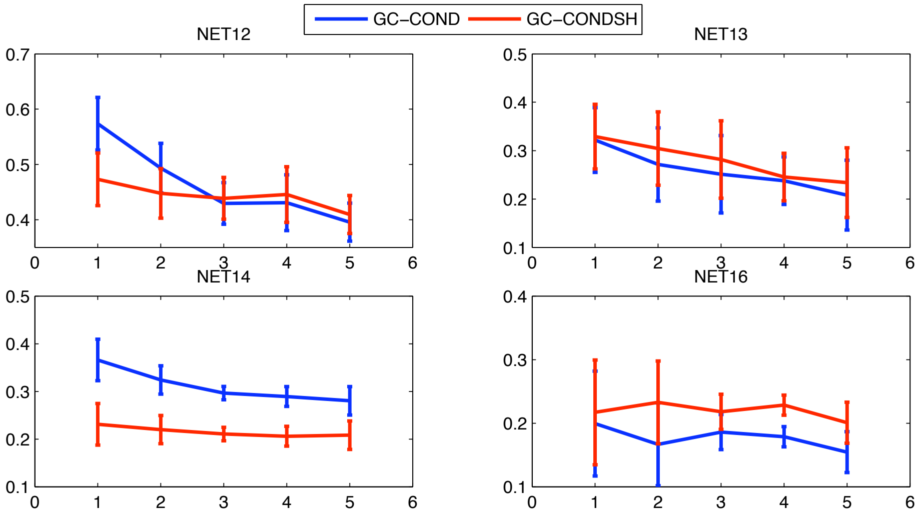


Figure 6.29: Comparison of GC-COND and GC-CONDSH models using number of shared edges that are captured correctly. x-axis corresponds to training data folds and y-axis corresponds to the F-score match between the shared edges in the true network pair versus the inferred network pair.

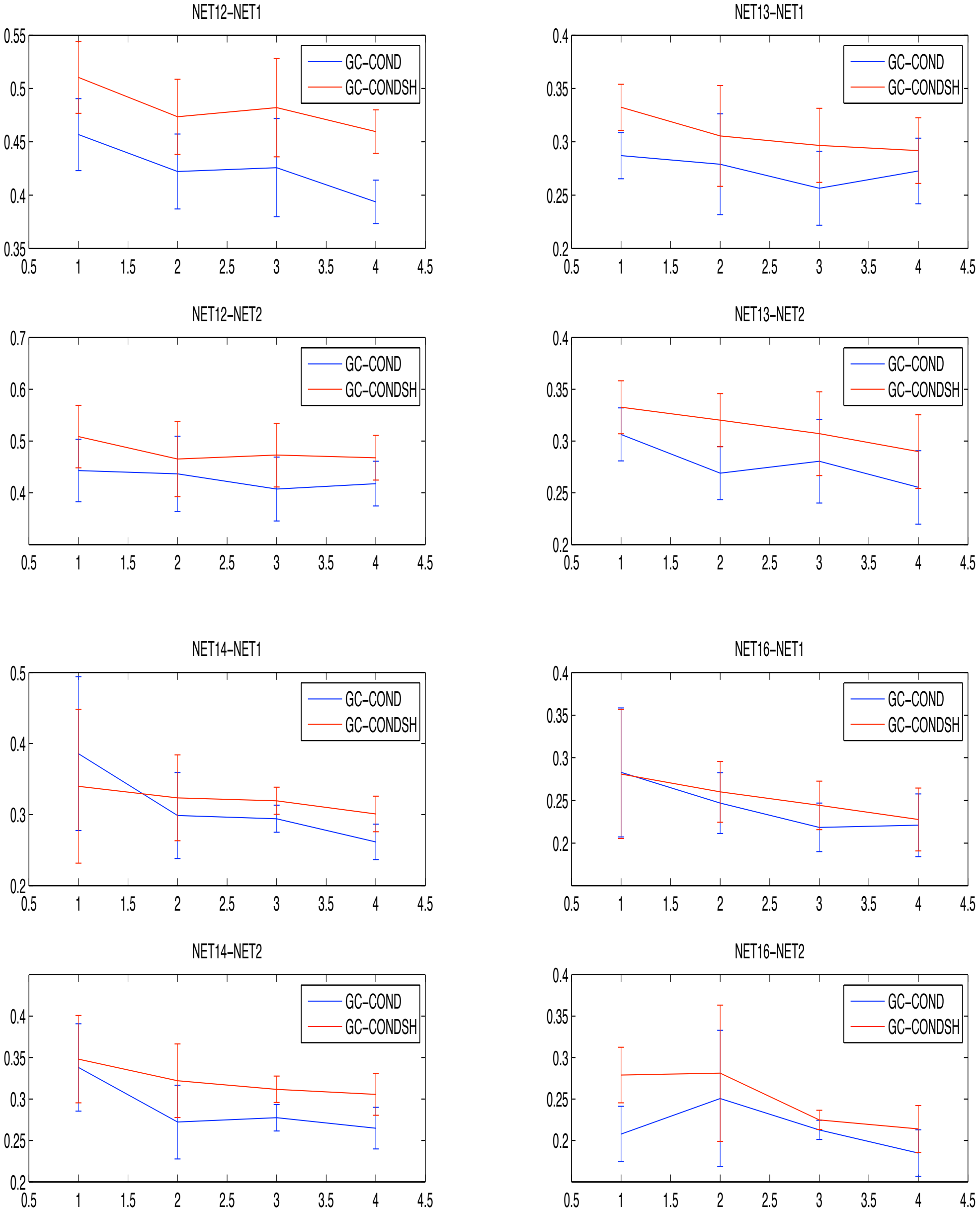


Figure 6.30: Comparison of GC-COND and GC-CONDSH models using stability of the network structure. x-axis is the number of folds of the training data. y-axis is the average F-score comparing a network from one fold to all other folds.

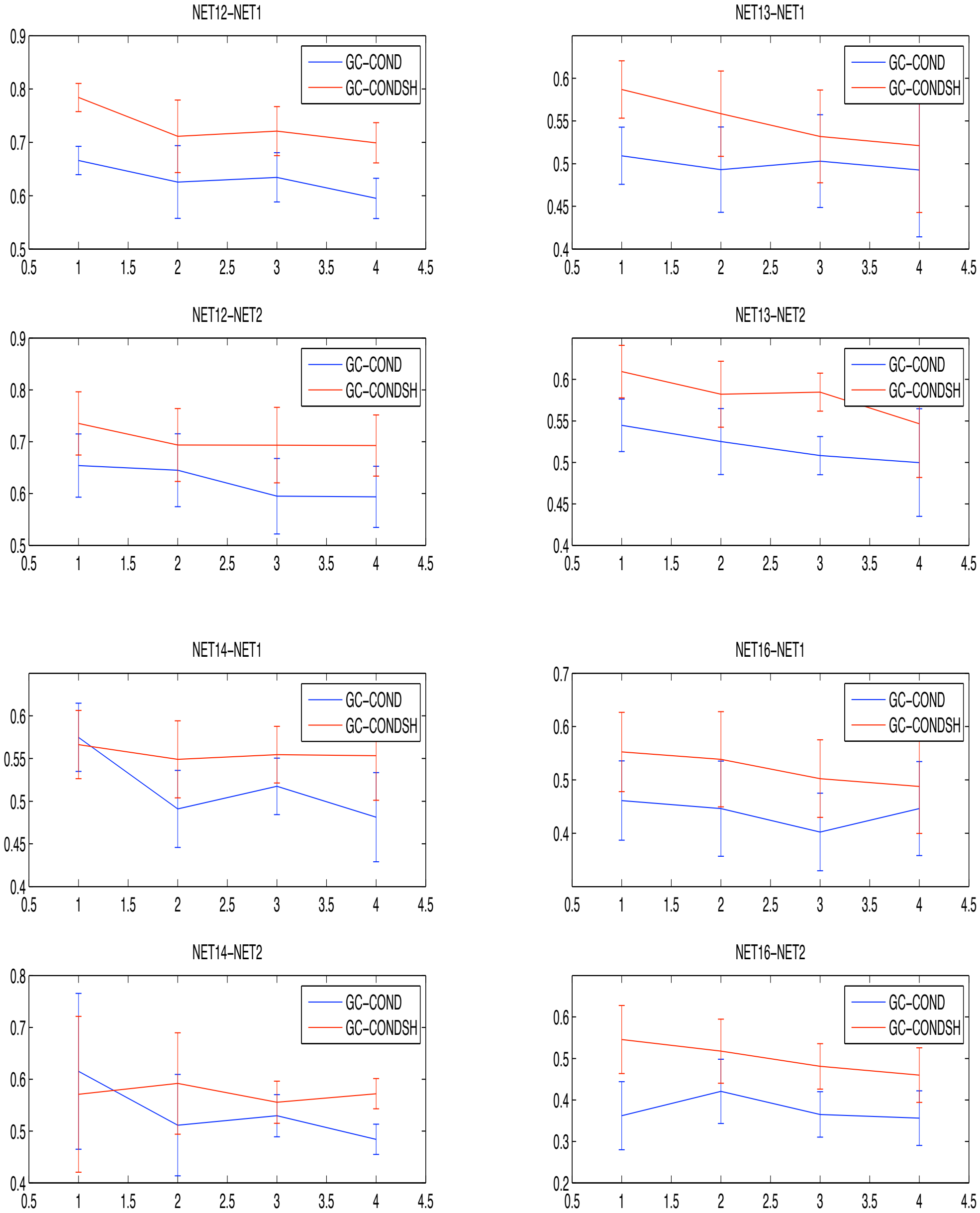


Figure 6.31: Comparison of GC-COND and GC-CONDSH models using stability of the network structure. We consider only the edges that are correctly captured across different folds.

Chapter 7

Application to condition-specific and species-specific networks

In this chapter, we apply our condition-specific network learning algorithms to real microarray data. We consider the problem of condition-specific network learning in the context of two biological questions: (a) what are the condition-specific and generic patterns in two different yeast cell populations, that are both under the same starvation stress, (b) what information can be captured at the network level about shared and specific features across different species. We address the (a) using the microarray datasets introduced in Chapters 5 and 6 measuring the gene expression of two yeast populations, quiescent and non-quiescent, isolated from glucose-starved stationary phase cultures. We address (b) using microarray data collected under diverse conditions from yeast and fly.

7.1 Learning condition-specific networks in yeast stationary phase

The main biological question we asked here was how two different cell populations behave, at the network level, in response to the same starvation stress. The two cell populations are quiescent (QUIESCENT) and non-quiescent (NON-QUIESCENT), isolated from glucose starvation-induced stationary phase cultures [5]. The two cell populations are in the same media but have differentiated physiologically and morphologically, suggesting that each population is responding differently. We learned networks using the two types of approaches used for condition-specific network inference. The first type which assumes the condition variable to be given, includes the Independent learner (INDEP) and the Network Inference with Pooling Data (NIPD) approaches. The second type which infers the condition variables includes Mixture of Undirected Graphs (MUG), Per-variable EM (PVEM), Graph component with conditional Gaussians (GC-COND) and GC-COND model with parameter tying (GC-CONDSH).

Because each array in the dataset was obtained from a single gene deletion mutant, the networks were constrained such that genes with deletion mutants connected to the remaining genes[1]. Hence, the INDEP networks described in this chapter are different from the ones described in Chapters 4 and 6. The inferred networks from the different methods were evaluated using information from Gene Ontology (GO) process, GO Slim [6] and transcriptional regulatory networks [90]. We also analyzed combinations of genes with deletions that were in the neighborhood of other non deletion genes.

[1] This is not a bi-partite graph because the genes with deletion mutants are allowed to connect to each other.

7.1.1 NIPD identified more biologically meaningful dependencies

To determine if one network was more biologically meaningful than the other, we examined the networks based on Gene Ontology (GO) slim category (process, function and location), transcription factor binding data and GO process, referred as GOSLIM, TFNET and GOPROC, respectively (Fig 7.1). Network quality was determined by the number of GOSLIM categories (or TFNET or GOPROC) with better coverage than random networks. Briefly, coverage allows us to measure network quality based on the number of annotation categories a network has better coverage than random networks with the same degree distributions (See Chapter 3 for details). We used all Gene ontology slim categories (process, function and location).

We found that the approaches that were given the condition variable (NIPD, INDEP) were much better than the approaches that had to infer the condition variable (MUG, PVEM, GC-COND, GC-CONDSH). This comes as no surprise because the datasets are labeled and there is no ambiguity about a data point about which condition it belongs. Thus the models that must infer the condition labels have a difficult learning job and do not perform as well.

INDEP and NIPD were equivalent on GOSLIM, with INDEP outperforming NIPD in QUIESCENT and NIPD outperforming INDEP on NON-QUIESCENT. NIPD outperformed INDEP with a larger margin than was outperformed on TFNET categories from NON-QUIESCENT. NIPD was consistently better than INDEP on GOPROC categories. GO Slim processes are described in detail in Appendix E, Table E.1.

We also evaluated the biological meaningfulness of the edges in each of the inferred networks using the average semantic similarity of genes connected in the networks [89] (Fig 7.2). A network that is more biologically meaningful has many more edges for a given semantic similarity than a network that is not as meaningful. With the exception of PVEM on NON-QUIESCENT, all networks had significantly more number of edges at a particular semantic similarity than random networks. Further, PVEM-inferred network had significantly more edges than random at high values semantic similarity. This suggests that all inferred networks captured many more biologically meaningful dependencies than similarly structured random networks.

We then compared the semantic similarity of the various methods among themselves (Fig 7.3). We found that on both populations, NIPD had the highest number of edges with a specific semantic similarity, indicating that NIPD overall was learning the best quality networks. Interestingly, on QUIESCENT majority of the models inferring the condition variable had more edges with high semantic similarity than INDEP. This suggests that despite the fact that these models had to infer the condition variable, the dependencies inferred by these algorithms are atleast as biologically meaningful as INDEP, to which the condition variable is given as input.

Finally, we estimated the percentage and the number of 1-n subgraphs that were significantly enriched in a GO process (Table 7.1) at a false discovery rate (FDR) < 0.05. An inferred network is considered good if it has many subgraphs that satisfy the FDR cutoff and also if these subgraphs constitute a high proportion of the total subgraphs considered. NIPD had almost twice the number of sub-graphs than INDEP on QUIESCENT, at comparable levels of subgraph proportion (86% vs 90%). Both MUG and PVEM performed at par with INDEP. The graph component models inferred slightly more subgraphs, which corresponded to the same proportion of subgraphs as NIPD.

NIPD also identified more subgraphs than INDEP on NON-QUIESCENT. PVEM performed poorly in both number and the percentage of subgraphs. Although GC-CONDSH and MUG had 100% of subgraphs that satisfied the FDR, the number of subgraphs was much less than of both NIPD and INDEP.

In summary, NIPD was the best algorithm for both QUIESCENT and NON-QUIESCENT. The other approaches performed similarity on QUIESCENT, but INDEP was better

Population	Method	# of Subgraphs	% of Subgraphs
QUIESCENT	INDEP	10	91%
	NIPD	19	86%
	MUG	9	91%
	PVEM	9	90%
	GC-COND	11	85%
	GC-CONSH	12	80%
NON-QUIESCENT	INDEP	9	82%
	NIPD	12	80%
	MUG	5	100%
	PVEM	3	50%
	GC-COND	6	75%
	GC-CONDSH	5	100%

Table 7.1: Number and percentage of subgraphs associated with a GO process using the quiescent and non-quiescent populations

than the approaches inferring the condition variable on NON-QUIESCENT. The fact that approaches that had to infer the condition variable performed as well as INDEP on QUIESCENT was promising.

7.1.2 Shared metabolic and regulatory processes in yeast stationary phase

We performed a more fine-grained analysis of the specific GO processes that the subgraphs were enriched in and complemented it with enrichment in the target sets of transcription factors (TFs). Using an FDR cutoff of 0.05, we identified many more subgraphs in the networks inferred by NIPD than by any of other approaches including INDEP to be enriched in a GO process or in targets of TFs (Tables 7.2, 7.3, Supplementary Table E.2 has the genes for each subgraph, Supplementary Figs E.1, E.2 show subgraphs for NIPD-inferred networks, E.3 and E.4 show subgraphs from INDEP-inferred networks). NIPD identified many more shared processes among two populations (aerobic respiration, mitochondrial electron transport, protein folding, fatty acid metabolism, ethanol metabolic process) than did any of the other approaches. Specifically shared processes identified by the other approaches were either identified by NIPD or were a child or parent term of a process identified by NIPD. (e.g. fatty acid beta oxidation found in GC-COND and INDEP is a child of fatty acid metabolism). The only difference was mono-carboxylic acid metabolic process which was found in both populations by GC-COND.

Analysis of the inferred networks on the basis of TFs indicated that all but the GC-CONDSH-inferred network had subgraphs involved in respiration to be also enriched in targets of HAP4, a global activator for respiration genes. The presence of HAP4 targets in both cell populations makes sense because both populations are experiencing glucose starvation and must switch to respiration for deriving energy. Additionally, some methods (NIPD, GC-COND, MUG) found the TFs, MSN2, MSN4, and HSF1, regulating subgraphs involved in protein folding. These TFs activate stress responses and are known to activate genes involved in heat, oxidative and starvation stress. NIPD also found targets of SIP4 in both populations. SIP4 is a transcriptional activator of gluconeogenesis [138], expressed highly in glucose repressed cells [86], and therefore would be expected to be present in both quiescent and non-quiescent cells.

We conclude that NIPD captures the shared information among the two populations the best, identifying more networks that were biologically relevant and informative about glucose starvation response than did any of the methods. Majority of the approaches inferring the condition variable identified more shared TFs than did INDEP, indicating that our mixture models were beneficial for capturing shared information.

7.1.3 Wiring differences capture population-specific starvation response

NIPD identified several processes associated exclusively with quiescent cells. This included regulatory processes (regulation of epigenetic gene expression, and regulation of nucleobase, nucleoside and nucleic acid metabolism) and metabolic processes (pentose phosphate shunt). These were novel predictions that highlight differences between these cells based on network wiring. INDEP identified only one population-specific GO process (response to reactive oxygen species[2] in NON-QUIESCENT). An INDEP identified subgraph specific to QUIESCENT (protein de-ubiquitination), was actually a subset of the NIPD-identified subgraph involved in epigenetic gene expression regulation, indicating that NIPD subsumed most of the information captured by INDEP.

Because NIPD was the best approach so far, we compared the approaches inferring the condition variables with respect to the NIPD findings. MUG identified protein folding and fatty acid oxidation to be specific to QUIESCENT, but it is likely that these are shared processes. GC-COND found regulation of gluconeogenesis to be specific to QUIESCENT, which is consistent with the result that NIPD identified TFs affecting gluconeogenesis in QUIESCENT. GC-COND additionally found pyruvate metabolism to be specific to QUIESCENT. Both GC-COND and GC-CONDSH found pentose phosphate shunt to be QUIESCENT-specific, which agrees with NIPD. GC-CONDSH found coenzyme A biosynthetic transport, which agrees with the presence of acetyl-CoA in QUIESCENT by NIPD.

NIPD-inferred QUIESCENT networks contained subgraphs enriched exclusively in targets of SKO1, and AZF1. Both of these are zinc finger TFs, with AZF1 protein expressed highly under non-fermentable carbon sources [129], and SKO1 which regulates low affinity glucose transporters [135], and are both consistent with the condition expe-

[2] This was present in NIPD, but did not satisfy our FDR cutoff

rienced by these cells. Unlike NIPD, which identified SIP4 to be associated with both populations, INDEP identified SIP4 only in QUIESCENT. However, as we describe in the previous section, it is more likely that SIP4 is involved in both QUIESCENT and NON-QUIESCENT populations. INDEP also found the TFs YAP7 and AFT2 exclusively in QUIESCENT and NON-QUIESCENT, respectively. YAP7 is involved in general stress response and would be expected to have targets in both QUIESCENT and NON-QUIESCENT. AFT2 is required under oxidative stress and is consistent with the overabundance of reactive oxygen species in NON-QUIESCENT population [3]. Both MUG and PVEM's list of TFs was a subset of NIPD.

GC-COND agreed with INDEP on HSF1, and on HAP4 and MSN2 with NIPD. Unlike both INDEP and NIPD, GC-COND identified SIP4 to be specific to Q. GC-CONDSH found all TFs associated with QUIESCENT, which is likely not correct. However GC-CONDSH found a novel TF, ASH1 and together with GC-COND, TF MCM1 to be important in QUIESCENT. ASH1 is involved in cell-fate determination localizing specifically in the daughter cells. This is consistent with the previous knowledge that the quiescent population is primarily composed of daughter cells [3]. The presence of both MCM1 and ASH1 is interesting, because both TFs are involved in specifying cell-fate, a key process for cellular differentiation. Experimental validation of these predictions may provide new insight into the cell-fate determination of Q versus NQ cells.

Overall, the NIPD inferred networks were of the highest quality capturing key differences and similarities in metabolic and regulatory processes, which are consistent with existing information about these cell populations [3, 5]. INDEP found many differences, but they are likely to be spurious. MUG captured a subset of processes and regulatory relationships captured by NIPD, and therefore had high precision, but low recall. PVEM did not have a significantly better advantage over other models inferring condition variables. The GC-COND and GC-CONDSH methods did have some discrepancies compared to NIPD, we found several novel processes and TFs that can provide new insight into starvation response in yeast.

	INDEP	NIPD	MUG	PVEM	GC-COND	GC-CONDSH
	17	22	8	10	11	15
aerobic respiration	QNQ	QNQ	QNQ	QNQ	QNQ	NQ
ammonium transport	QNQ	QNQ	NQ	QNQ	QNQ	QNQ
nitrogen utilization	QNQ	QNQ	NQ	QNQ	Q	QNQ
carboxylic acid biosynthetic process	Q	Q	Q	Q		NQ
organelle ATP synthesis coupled electron transport		QNQ	QNQ	QNQ	QNQ	QNQ
mitochondrial electron transport, ubiquinol to cytochrome c	NQ	QNQ	NQ		QNQ	NQ
protein folding	NQ	QNQ	Q	Q		
oxidative phosphorylation	QNQ	QNQ		QNQ	Q	
carnitine metabolic process	Q	NQ				NQ
fatty acid beta-oxidation	QNQ				QNQ	
pentose-phosphate shunt		Q				Q
polyamine catabolic process	NQ	QNQ				
beta-alanine biosynthetic process	NQ	QNQ				
pyruvate metabolic process				NQ	Q	
energy derivation by oxidation of organic compounds		QNQ				QNQ
negative regulation of biosynthetic process						Q
coenzyme A biosynthetic process						Q
carboxylic acid transport						Q
ethanol metabolic process		QNQ				
pentose metabolic process		Q				
acetyl-CoA metabolic process		Q				
pentose-phosphate shunt, oxidative branch					Q	
fatty acid metabolic process		QNQ				
protein deubiquitination	Q					
fatty acid oxidation			Q			
monocarboxylic acid metabolic process						QNQ
regulation of gene expression, epigenetic		Q				
regulation of gluconeogenesis					Q	
regulation of nucleobase, nucleoside, nucleotide and nucleic acid metabolic process		Q				
response to metal ion		Q				
tricarboxylic acid cycle intermediate metabolic process	Q					
alcohol metabolic process						NQ
amine catabolic process				NQ		
ATP synthesis coupled proton transport	NQ					
generation of precursor metabolites and energy	NQ					
ion transport		NQ				
mitochondrial electron transport, succinate to ubiquinone	NQ					
NADH regeneration		Q				
N-terminal protein myristoylation				NQ		
regulation of DNA replication						NQ
response to reactive oxygen species	NQ					
septin ring assembly					NQ	

Table 7.2: Specific GO biological processes identified by each method in the quiescent (Q), non-quiescent (NQ) or both populations (QNQ).

	INDEP	NIPD	MUG	PVEM	GC-COND	GC-CONDSH
	8	8	5	4	5	5
HSF1	NQ	QNQ	QNQ	Q	NQ	Q
HAP4	QNQ	QNQ	QNQ	QNQ	QNQ	
MSN2	NQ	QNQ	QNQ		QNQ	Q
SIP4	NQ	QNQ	Q	NQ	Q	
MSN4	NQ	QNQ	QNQ			Q
MCM1					Q	Q
HAP2		Q		QNQ		
AZF1		Q				
HAP1	Q					
ASH1						Q
SKO1		Q				
YAP7	Q					
AFT2	NQ					

Table 7.3: Transcription factors with targets enriched in inferred subgraphs in the quiescent (Q), non-quiescent (NQ) and both (QNQ) populations.

7.1.4 NIPD identified several deletion combinations

The microarrays used in this study measured expression profile of single gene deletions that were previously identified to be highly expressed at the mRNA level in stationary phase. We constrained the inferred networks to identify neighborhoods of genes comprising only the genes with deletion mutants, allowing us to identify combinations of such deletion mutants and their targets. Such combinations can be validated in the laboratory to verify cross-talk between pathways. We found that NIPD-inferred networks contained significantly more deletion combinations compared to random networks for both the quiescent and non-quiescent populations (p-value $< 3E$-7), which was not the case for the any of the other inferred networks.

We performed a more stringent analysis of the knock-out combinations by comparing the semantic similarity between a gene, g_i and the set of knock-out genes that are connected to it, $\mathbf{K}_i$. We assumed GO process terms for the set $\mathbf{K}_i$ to be the union of all terms associated with the genes, $g_j \in \mathbf{K}_i$. We then computed the semantic similarity between the term set associated with gene g_i and the union of terms associated with $\mathbf{K}_i$ and assessed statistical significance from a background distribution estimated from random networks with the same degree distributions.

This stringent analysis of the knock-out combinations identified several double knock-out and target gene candidates (Table 7.4). We also found more deletion combinations in NON-QUIESCENT compared to QUIESCENT. This is consistent with the identification of many more mutants affecting non-quiescent than quiescent cells [5]. In QUIESCENT, we found three genes that were all likely down-stream targets of a COX7-QCR8 double knock-outs, all involved in the cytochrome-c oxidase complex of the mitochondrial inner membrane. Other deletion mutant combinations were involved in mitochondrial ATP synthesis and ion transport. Many of these genes have been shown to be required for quiescent/non-quiescent cell function, viability and survival [94, 5]. In NON-QUIESCENT, we found several knock-out combinations involved in oxidative phosphorylation, aerobic respiration etc, including a novel combination, YMR31 and QCR8, connected to TPS2. All three genes are found in the mitochondria, which play a critical and complex role in starved cells, but the exact mechanisms are not well-understood. Experimental analysis of this triplet can provide new insights into the role of mitochondria in glucose-starved cells.

7.2 Condition-specific networks for learning inter-species networks

The second application of our condition-specific network learning framework asks what shared and specific subnetworks can be identified among orthologous genes from different species. In this application we treat the condition variable as different species. We consider two species: yeast (*S. cerevisiae*) and fly (*D. melanogaster*). This question of finding

Deletion combinations	Downstream effect	Process
	QUIESCENT	
COX7*, QCR8	COX13, QCR6, QCR9	ATP synthesis coupled electron transport
ADY2*, CTA1*	ATO3	ion transport
ETR1*, ACS1	AYR1	carboxylic acid metabolic process
	NON-QUIESCENT	
ATP12, SDH2*	ATP16	oxidative phosphorylation
YMR31, QCR8	TPS2	trehalose bio-synthesis, mitochondrial function
ADY2*, YAL054C	ATO3	ion transport
NQM1, QCR8	COX13	aerobic respiration
COX7*, QCR8	QCR9	electron transport
SIP18, YGR236C	AZR1	response to stimulus
ETR1*, ADH2	LSC2	energy derivation by oxidation

Table 7.4: Knock-out combinations identified by NIPD in the quiescent and non-quiescent populations. Genes with * have been shown to have a phenotype in stationary phase or in quiescent and non-quiescent cells in [94, 5].

conserved interactions across species is an important question of comparative genomics and has been addressed at the level of co-expression by Stuart *et al.* [131] and Bergmann *et al.* [13]. However, we address the problem in a more general setting where we identify conserved higher-order dependencies among the two organisms.

Following Stuart *et al.*, we consider only genes that are orthologs of each other, that is, genes that share a common ancestor species and are likely performing the same function in both organisms. These orthologs are identified by high sequence similarity between genes of one organism and genes of another organism. Orthologs are the individual genes that are conserved across different organisms. By learning networks across species using orthologs, we identify functional relationships that are conserved across organisms.

7.2.1 Data processing

We used the datasets assembled by Stuart *et al.* [131], which consist of microarrays from both organisms measuring genome-wide expression profile of yeast and fly genes under different environmental conditions of yeast or different life-cycle stages of fly. Only genes with orthologs were considered in both datasets. An orthologous pair of genes was referred as *meta-gene*. A random variable in our framework represents a meta-gene in both organisms. It is possible that a gene in one organism has multiple orthologous hits. In that case, we consider the gene with the maximal number of microarray measurements. Thus we have a single pair of orthologs for each gene. We had a total of 923 genes in our datasets after this processing. After network learning, we expand the network to include multiple hits by adding edges among all orthologous pairs of a gene, if the pair in the inferred network is predicted to be connected. Specifically, assume gene A_{FLY} has corresponding orthologous genes in yeast A_{YEAST1} and A_{YEAST2}. Assume we selected A_{YEAST1} during network inference. After network inference is complete, if A_{FLY} is inferred to be connected to A_{YEAST1}, we introduce an additional edge between A_{FLY} and A_{YEAST2}.

7.2.2 Results

We selected four models for learning networks for the yeast and fly species. Two of these are INDEP and NIPD approaches, which assume condition variable is given. The remaining two are the GC-COND and GC-CONDSH models, which infer the condition variable. We excluded PVEM from this analysis because our simulated data experiments indicated that the graph component models had better performance overall compared to PVEM. We excluded MUG because application to yeast stationary phase suggested that although the structures were high quality, they were very sparse.

The goals of our analysis of the inferred networks were (a) to evaluate the quality of the networks, and (b) to assess the extent of organism-specific and conserved relationships.

We address (a) using the GO process enrichment of subgraphs in the inferred networks and by comparing against a meta-gene co-expression network inferred by Stuart *et al.*, and (b) by identifying the shared and specific processes in which the subgraphs from each organism-specific network are enriched.

Graph component models identify a high proportion of biologically meaningful dependencies

Similar to our analysis of networks inferred from yeast stationary phase, we obtained the proportion of 1-n subgraphs that were enriched in a GO process at an FDR < 0.05 (Table 7.5). We found that on the YEAST network GC-CONDSH was better than INDEP (high number and proportion of subgraphs). While GC-COND too had a high proportion of subgraphs, the number of subgraphs was lower than the other methods. We note that unlike the yeast stationary phase, where each method despite its high proportion identified very few subgraphs, in this case the methods do identify a non-trivial number of subgraphs ($>$100). Thus for this dataset, we can use the proportion to be a good indicator of method superiority. On FLY also, the graph component models had a better performance than INDEP. The fact that GC-CONDSH is able to learn better networks than GC-COND for this application is promising because it demonstrates the advantages of fine-grained parameter tying-based information sharing.

We further compared the structure of our inferred networks with the meta-gene network inferred by the Stuart *et al.* study which considered conserved co-expression relationships (Fig 7.4). Because the networks learned by our approaches impose an upper limit ($k = 8$) on the size of the neighborhood of each gene, we constrained the meta-gene network by considering only the best $k = 8$ genes connected to a gene. Because this produces inconsistencies in the network structure, that is gene A may be connected to gene B, but not vice-versa, we apply an AND or OR processing to each gene's neighborhood to produce consistent graphs. In the AND processing, if A is in B's neighborhood, but not vice-versa, we delete A from B's neighborhood. In the OR processing, if A is in B's neighborhood, but not vice versa, we add B to A's neighborhood. This type of post-processing is the same as described in Chapter 4, which is required to make the network structures consistent. We refer to the complete yeast-fly co-expression network inferred by Stuart *et al.* as the UNCONST, the constrained network with OR post-processing as KNN-OR[3] and with AND processing as KNN-AND.

Both NIPD and INDEP had more agreement with the existing meta-gene network than GC-COND and GC-CONDSH on YEAST, regardless of whether we were comparing the neighbor size-constrained networks or the unconstrained networks. However, GC-CONDSH had more agreement than GC-COND. In contrast to YEAST, GC-COND and GC-CONDSH-inferred FLY networks had a better match than both INDEP and NIPD when considering the KNN-OR and KKN-AND networks. GC-COND and GC-CONDSH had a better match than INDEP and were similar to NIPD, on the unconstrained network.

Overall, we found the GC-COND and GC-CONDSH models to perform as well or better than the INDEP model in the majority of the comparisons. This suggested that when constructing networks across species, models that infer the condition variable are more beneficial than models that do not perform any type of sharing. Further, GC-CONDSH performs better than GC-COND suggesting that cross-species data contain more subtle sharing patterns that can be captured by models incorporating complex sharing patterns via fine-grained parameter-tying.

Biological processes conserved and unique to different species

We compared the inferred networks to identify the shared and unique processes in each organism Table 7.6. We first consider the shared processes between yeast and fly. Interestingly, we found that in this case GC-CONDSH identified the largest number of shared

[3] KNN: K nearest neighbors

Species	Method	# of Subgraphs	% of Subgraphs
YEAST	INDEP	170	61%
	NIPD	213	71%
	GC-COND	124	89%
	GC-CONSH	185	86%
FLY	INDEP	147	78%
	NIPD	176	81%
	GC-COND	120	79%
	GC-CONDSH	138	80%

Table 7.5: Number and percentage of subgraphs associated with a GO process using the yeast and fly networks.

processes (18), followed by NIPD (15), GC-COND (10) and INDEP (8).

Examination of the individual processes from each method indicated that NIPD identified several metabolic processes (glycine catabolic process, fatty acid oxidation, pyruvate metabolism) and ribosome functions (rRNA processing and RNA elongation) and energy related processes (oxidative phosphorylation). Existing work of Stuart *et al.* identified ribosome biogenesis conserved between yeast and fly. We additionally identified fatty acid, pyruvate metabolism and IMP biosynthesis to be conserved between yeast and fly. Pyruvate metabolism and fatty acid metabolism have been implicated to play a central role in alcoholism studies in fly [103]. Because yeast is such a well-studied model organism, the conservation of this process in yeast and fly allows us to extrapolate the knowledge available in yeast to fly and potentially further our understanding in alcoholism in humans. IMP, which means inosine monophosphate is the end-product of purine bio-synthesis, the pathway responsible for the production of building blocks of RNA and DNA molecules, and is consistent with being conserved in yeast to fly. All processes identified by INDEP were detected by NIPD.

GC-COND and NIPD had some overlap (6) including ribosome function and ubiquitin catabolism and energy derivation by oxidation. GC-COND also identified protein amino acid phosphorylation. Phosphorylation is a key mechanisms of activating and inhibiting proteins and enzymes and therefore it is expected to be conserved across a diverse species.

GC-CONDSH and NIPD also had several processes (9) in common (fatty-acid beta oxidation, pyruvate metabolism and IMP bio-synthesis). However, it included several novel processes (GTPase mediated signal transduction, phospholipid biosynthesis, protein folding) that were conserved between fly and yeast. The specific yeast genes in subgraphs enriched in GTPase mediated signal transduction were involved in signaling pathways in nutrient limiting conditions and other pathways involving rapamycin and MAPK signaling pathways. The fly-specific genes in the enriched subgraphs were involved in Ras signal transduction and growth regulation, which are all important for cell proliferation and differentiation. The identification of these conserved processes makes biological sense because these processes are fundamental to system survival at the cell, tissue and the entire organism.

Overall, our results suggested that a method that does not do any type of sharing (INDEP) missed out of biological processes that are both biologically meaningful and important to identify. Although the methods with some type of sharing did not completely agree on the set of shared processes, they all made biological sense.

We also identified processes to be specific to each species. We note that the identification of true differences across species is tricky. Because fly is not as well annotated as yeast, identifying a particular process in yeast, but not in fly, may not necessarily mean the process is yeast-specific. In particular, there are two reasons why an identified process may not necessarily be yeast-specific: (a) the process does exist in fly, but the fly genes involved in the process are not in our dataset, (b) no fly gene has yet been identified for the particular process. To address (a), we consider only those processes to be truly yeast-specific if no fly gene has been annotated with that process. While (b) is harder to address, we make the assumption that if there is no gene annotated yet with a process, then the process does not exist in fly. Further, we consider only those processes that are found by at least three methods. A similar strategy was used to identify fly-specific process. However, because yeast is far more studied than fly, the processes we predictions of fly-specific processes are more likely to be true than the predictions of the yeast-specific processes.

Based on the above strategy, we found most of the yeast-related processes were involved in glucose metabolism (pentose-phospate shunt, trehalose catabolism, regulation of cAMP biosynthesis) and filamentous growth. The fly-specific processes included animal-specific processes (heme transport, blood-brain barrier, oocyte fate determination). Although these processes capture meaningful species-specific behavior, we did have some surprises. For example, even though small GTPase mediated signal transduction was found to be a conserved process, we found subgraphs that were specific to yeast involved in the negative regulation of the cAMP biosynthetic process. It is possible that this difference is due to the incompleteness in the fly annotation, because the fly ortholog of the yeast genes involved in the negative regulation of cAMP, Neurofibromin, is involved in a variety of regulatory processes, including RAS signal transduction. However, the fly-specific processes and the remaining fungi-specific processes agree with biological intuition and are likely to be true positives.

7.3 Suitability of methods for different condition-specific network learning problems

In this section, we consider the suitability of different algorithms for learning condition-specific networks depending upon the dataset attributes. This analysis is based on both the results on simulated data from Chapter 6 as well as results on real data from this Chapter. The selection of an algorithm for a dataset depends upon the attributes of the dataset (Fig 7.5). We describe a dataset using three attributes: (a) condition variable, (b) similarity in the underlying networks, (c) dataset complexity. The condition variable can take the values: known and unknown. The expected similarity in the underlying networks can take values: high similarity and low similarity. The dataset complexity can take values: simple and complex.

The first two attributes are self-explanatory. The dataset complexity attribute requires more explanation. This attribute provides a coarse way of describing the experimental conditions under which the dataset is generated. In particular, an experimental condition is said to be simple if we vary only the condition variable but make all measurements under the different condition values in a similar way. Such simple datasets are obtained, for example, when we have the same cell population under different stress conditions, or we have two different cell populations under the same environmental stress as in the quiescent and non-quiescent population. An experimental condition is said to be complex, if the data under the different values of the conditions are not collected in a similar way. These cases arise when the data is generated from different organisms or from different laboratories.

We realize that this may be an overly simplified way of describing datasets, but it allows us to provide guidelines to users of our algorithms to determine the applicability of our different methods based on their knowledge of the datasets. Because the network similarity attribute may not be known beforehand, we allow this attributes to take the *unknown* value and also describe the most suitable method under this scenario.

When the dataset is simple, and the condition variable is observed we suggest the use of NIPD, MUG or GC-COND. We suggest the MUG or GC-COND models to allow some amount of freedom for handling complex sharing patterns. We suggest the use of both models because while the MUG model had good performance on the simulated data, it found very few sub-networks on microarray data from yeast stationary phase. In contrast, the GC-COND model was beaten on a few simulated data cases, but retrieved many more biologically meaningful dependencies on the microarray data. Thus both models have strengths that may prove beneficial under simple dataset cases. When the dataset is complex, GC-CONDSH or PVEM models should be selected. Although, we do not have results of the PVEM models on the species-specific networks, the fact that it works well on the NET12-79 networks suggests that both PVEM and GC-COND models are good candidates of capturing complex sharing patterns. However, the PVEM model has the longest training time and therefore should be used if the networks are small in size and time is not a major constraint. If additionally the condition variable is observed, NIPD model is a good candidate. The majority of our simulated datasets are examples of simple datasets. However, even in these cases, if the network similarity is low and we want to find similarities, GC-CONDSH model should be selected.

7.4 Conclusion

The problem of learning condition-specific networks is an important, yet challenging problem of systems biology. One challenge is the complexity of the sharing patterns across conditions, which depends upon the definition of a condition. In this chapter, we apply our different approaches for learning condition-specific networks to address two biologically motivated questions that each present a different instance of the condition-specific network learning problem. In the first, we ask how two different yeast populations behave at the network level under the same stress. In the second, we ask what types of functional relationships are conserved among orthologous genes of two distant species: yeast and fly. The application of our network learning approaches identified shared and unique aspects of the networks that agree with existing biological knowledge and also include novel discoveries.

Application of our approaches to understand glucose starvation stress response in two morphologically dissimilar yeast populations, identified several strengths and weaknesses of the different algorithms. The methods that infer the condition variables are very general that can be applied to any problem of condition-specific network learning. However, the generality comes at the computational overhead of inferring the condition variables, which for the yeast stationary phase data was too high. In particular, we found that the NIPD approach to which we specify the condition variable (NIPD) and which incorporates sharing by pooling data, is significantly better than the approaches that infer the condition variable. Compared to INDEP, which does not perform any sharing, these models captured shared information as well or better, suggesting that at least these models are benefitted by sharing information. However, the excessive long training times of these models are hardly justified by the modest improvements in capturing shared information. Comparison of the models inferring the condition variables among each other showed that although MUG had a lot of agreement with NIPD, suggesting high precision, it identified very few processes producing high false negatives. In contrast, the graph component models did have some disagreements with NIPD, but the differences were often biologically meaningful.

One of the strengths of an approach that shares data (NIPD) in comparison with that an approach that does not (INDEP), was the ability to identify more complex interactions such as pairs of gene deletions and downstream targets using data from individual gene deletions. Amazingly, several of these gene deletions are already known to have a phenotypic effect on stationary phase cultures and often on quiescent or non-quiescent cells [5, 94]. These predictions are therefore good candidates for future experiments using double deletion mutants, and are a drastic reduction of the space of possible combinations of sixty-nine single gene deletions. Identification of population-specific malfunctions in signaling pathways via experimental analysis of these multiple deletions can provide new insight into aging and cancer studies using yeast stationary phase as a model system.

Application of our approaches to learn species-specific networks, not only demonstrated another formulation of condition-specific network learning, but also identified an example dataset, which is complex enough to justify the use of more general models inferring the condition variables. In particular, graph component models (GC-COND and GC-CONDSH) identified many more similarities than INDEP and often NIPD. Interestingly, the GC-CONDSH model was able to identify key shared processes and also had better agreement with an existing meta-gene co-expression network, than GC-COND. This supported our conjecture in the last chapter that benefits of the GC-CONDSH are not apparent because our simulation datasets did not have complex shared patterns, which would be captured by the GC-CONDSH model.

Learning networks across different organisms (species-specific networks) is more challenging than learning networks across two conditions from the same organism. Because the organisms are so different from each other, identification of similarities and differences is more complicated than learning networks from the same organism. The fact that any model with some type of sharing, regardless of whether the condition variable is hidden or not, is better than a model that learns networks from the conditions independently, is suggestive of the complex sharing patterns in the species-specific networks. This is further supported by the improved performance of the GC-CONDSH model, which additionally performs fine-tuned sharing via parameter tying. However, these are preliminary results, which are susceptible to the relative incompleteness in the amount of known information for the different species, making validation of the true differences versus differences due to the absence of knowledge difficult. Further experimentation and validation is a direction of future research.

In summary, the two datasets analyzed in this chapter represent two points on a spectrum of datasets of varying shared pattern complexity. The yeast stationary phase dataset is one where the networks are likely to be highly similar because of the shared global starvation stress. The species-specific dataset is one where the networks, coming from different organisms, are very different with complicated sharing patterns. For all datasets, models with some type of sharing are better than learning networks for the conditions independently. Our preliminary analysis on the species-specific networks suggests that GC-CONDSH model is suitable for capturing such complexities in the shared information. On datasets where the underlying networks are highly similar, or where the condition variable has a similar influence on the entire network, the benefits of a general model inferring the condition variable are likely outweighed by the run time and parametric complexity of the models. In such datasets, models that accept the condition variable as input and incorporate sharing across conditions are more beneficial than more general methods. However, in datasets similar to the species-specific datasets, where different parts of the network are influenced differently by the condition variable, the mathematical complexity is justified, and often, needed to learn better condition-specific networks.

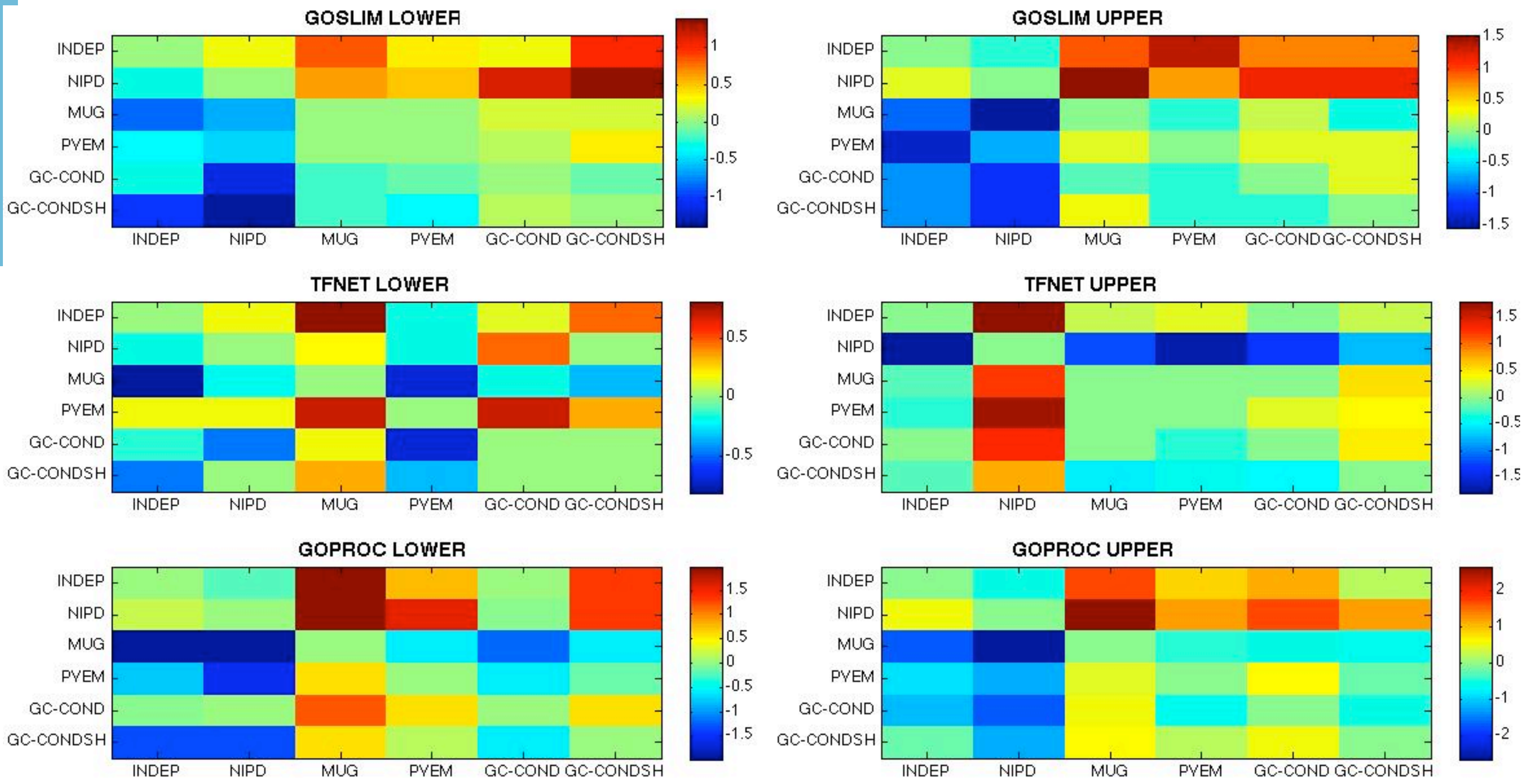


Figure 7.1: Coverage analysis of different annotation categories. Each cell i, j specifies the log of the ratio of the number times algorithm i beats the algorithm j to the number of times algorithm j beats algorithm i. i and j correspond to row and column respectively. The more red a cell the more likely is i better than j. The more blue the more likely is j better than i. LOWER corresponds to the quiescent cells and UPPER corresponds to non-quiescent cells.

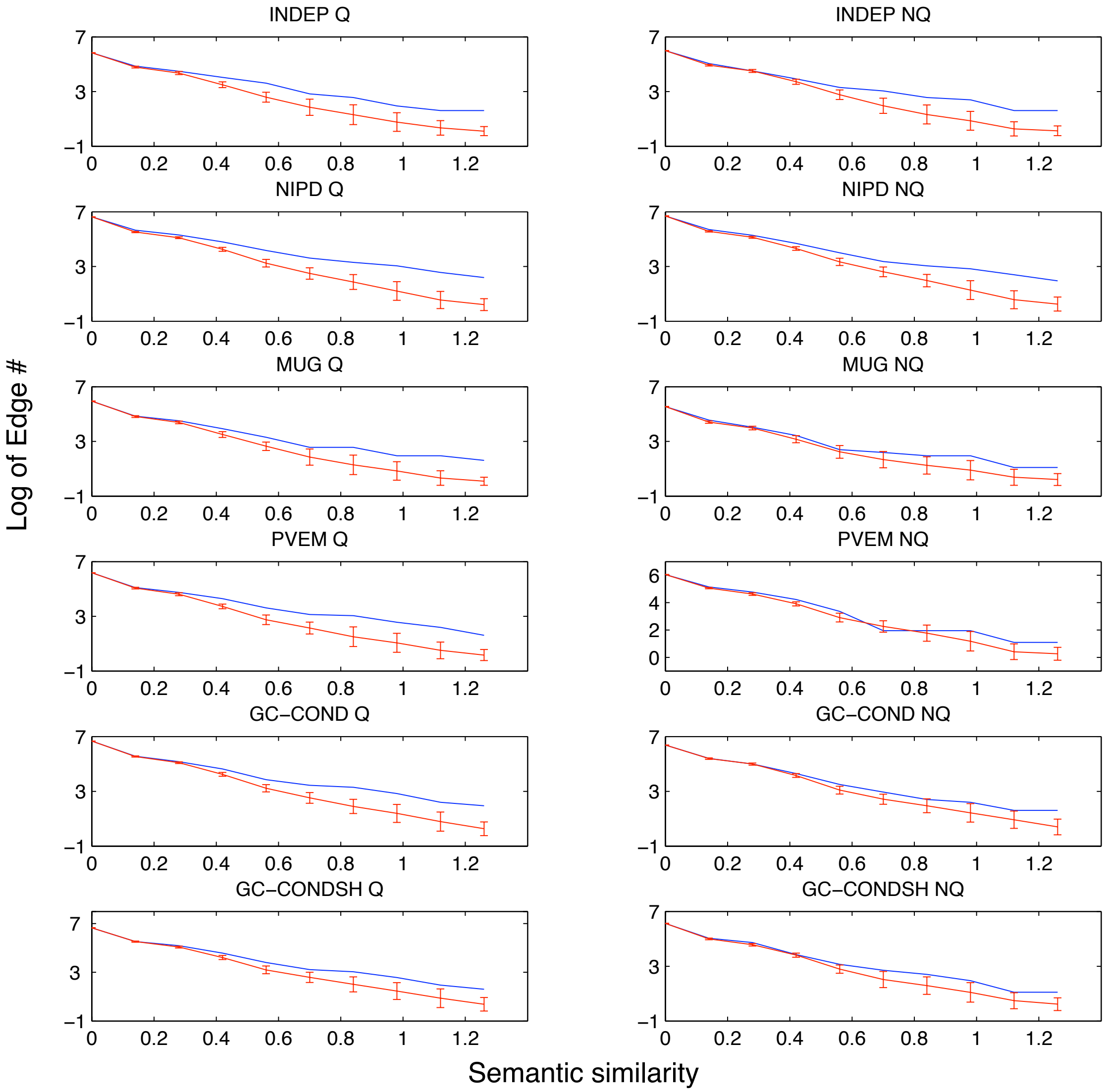


Figure 7.2: Number of inferred edges as a function of semantic similarity. The blue line indicates the number of edge in the inferred network and red line is the mean number of edges in random networks with the same degree distribution as the inferred network. Q: Quiescent, NQ: Non-quiescent.

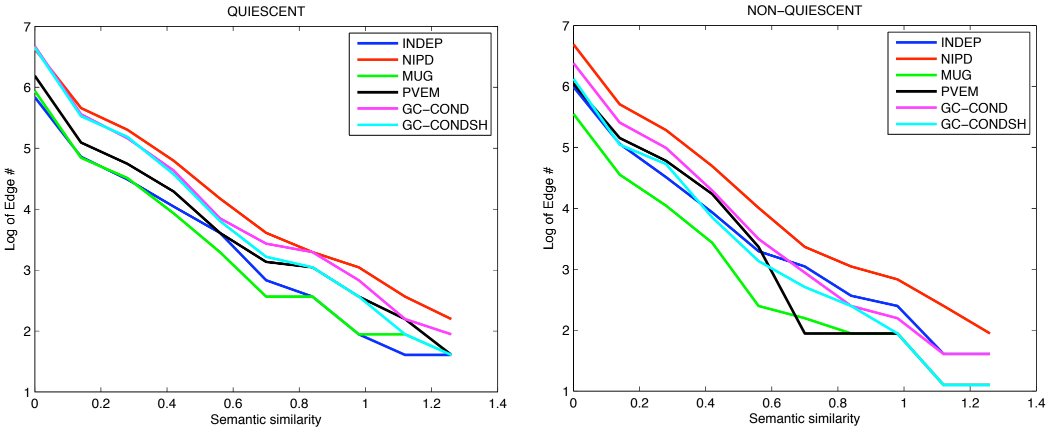


Figure 7.3: Semantic similarity of inferred graphs from different methods.

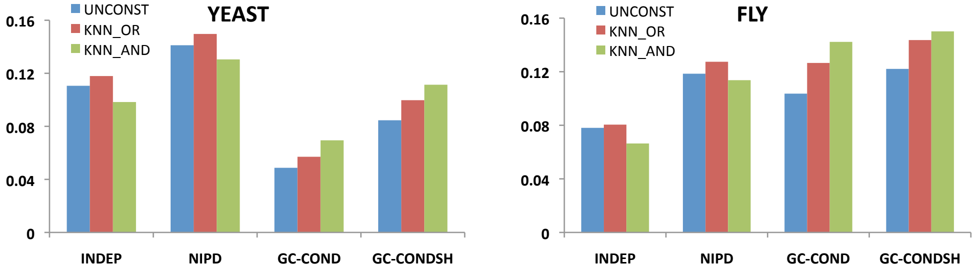


Figure 7.4: Comparison of the inferred networks by our different condition-specific network learning approaches against the networks inferred by the Stuart *et al.* study.

	INDEP	NIPD	GC-COND	GC-CONDSH
ATP synthesis coupled proton transport	FLY	YEAST-FLY	YEAST	YEAST-FLY
'de novo' IMP biosynthetic process	YEAST-FLY	YEAST-FLY	YEAST-FLY	YEAST-FLY
DNA replication initiation	YEAST-FLY	YEAST-FLY	YEAST-FLY	YEAST-FLY
fatty acid beta-oxidation	YEAST-FLY	YEAST-FLY	FLY	YEAST-FLY
glutamine family amino acid biosynthetic process	YEAST	YEAST-FLY	FLY	FLY
glycine catabolic process	YEAST	YEAST-FLY	YEAST-FLY	YEAST-FLY
mitochondrial electron transport, ubiquinol to cytochrome c	YEAST-FLY	YEAST-FLY	FLY	YEAST-FLY
mitotic chromosome condensation	FLY	FLY	FLY	YEAST-FLY
oxidative phosphorylation	YEAST-FLY	YEAST-FLY	FLY	FLY
pre-replicative complex assembly	YEAST-FLY	YEAST-FLY	FLY	YEAST-FLY
protein amino acid phosphorylation	FLY	FLY	YEAST-FLY	FLY
pyruvate metabolic process	FLY	YEAST-FLY	YEAST	YEAST-FLY
RNA processing	FLY	YEAST	YEAST-FLY	YEAST-FLY
rRNA processing	YEAST	YEAST-FLY	YEAST-FLY	YEAST
sulfur amino acid biosynthetic process	FLY	FLY	FLY	YEAST-FLY
transcription from RNA polymerase II promoter	YEAST	YEAST	FLY	YEAST-FLY
ubiquitin-dependent protein catabolic process	YEAST-FLY	YEAST-FLY	YEAST-FLY	YEAST-FLY
tRNA aminoacylation for protein translation		YEAST-FLY	YEAST-FLY	YEAST-FLY
energy derivation by oxidation of organic compounds		YEAST	YEAST-FLY	
cellular protein metabolic process	YEAST-FLY	YEAST-FLY		
vesicle docking during exocytosis			YEAST	YEAST-FLY
phospholipid biosynthetic process			YEAST	YEAST-FLY
protein folding	YEAST			YEAST-FLY
small GTPase mediated signal transduction				YEAST-FLY
cellular biosynthetic process			YEAST-FLY	
RNA elongation		YEAST-FLY		
ceramide biosynthetic process	YEAST	YEAST	YEAST	YEAST
negative regulation of cAMP biosynthetic process	YEAST	YEAST	YEAST	YEAST
negative regulation of ubiquitin-protein ligase activity during mitotic cell cycle	YEAST	YEAST	YEAST	YEAST
pentose-phosphate shunt, oxidative branch	YEAST	YEAST	YEAST	YEAST
septin checkpoint	YEAST	YEAST	YEAST	YEAST
signal transduction during filamentous growth	YEAST	YEAST	YEAST	YEAST
trehalose catabolic process	YEAST	YEAST	YEAST	YEAST
UDP-glucose metabolic process	YEAST	YEAST	YEAST	YEAST
protein-RNA complex assembly	YEAST	YEAST	YEAST	
heteroduplex formation	YEAST		YEAST	YEAST
cell wall chitin biosynthetic process		YEAST	YEAST	YEAST
methylglyoxal catabolic process to D-lactate		YEAST	YEAST	YEAST
secretory pathway	YEAST	YEAST		YEAST
establishment of blood-brain barrier	FLY	FLY	FLY	FLY
glutamate catabolic process to 2-oxoglutarate	FLY	FLY	FLY	FLY
heme transport	FLY	FLY	FLY	FLY
cellular biopolymer biosynthetic process	FLY	FLY		FLY
germarium-derived oocyte fate determination	FLY		FLY	FLY
oocyte microtubule cytoskeleton polarization	FLY		FLY	FLY
oxidation reduction	FLY		FLY	FLY

Table 7.6: Biological processes specific to yeast (YEAST) or fly (FLY) or shared between the two species (YEAST-FLY). The leftmost column is the process and the remaining columns are for each of the algorithms. For processes that are specific to each population, we consider only those processes that are identified by at least three of the four methods.

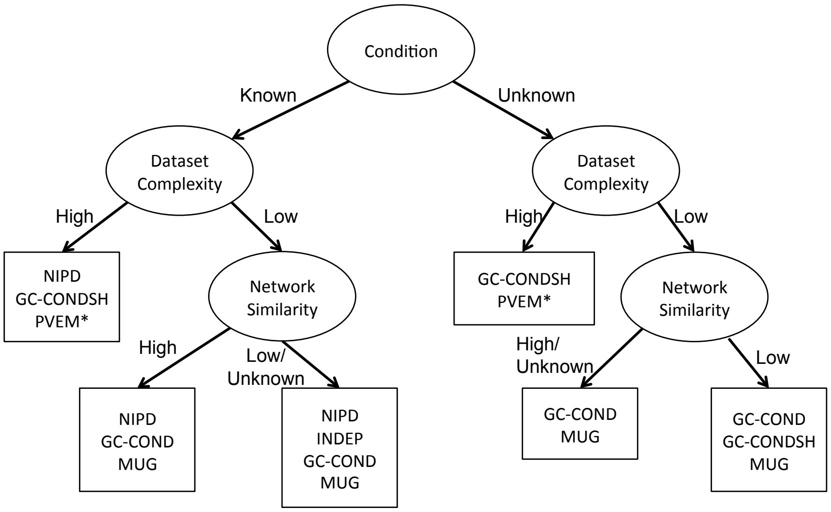


Figure 7.5: Decision tree for selecting different models based on dataset attributes. PVEM has a * because our placement of PVEM in the decision tree is based only on simulated data.

Chapter 8

Conclusions

Inference and analysis of cellular networks has been one of the cornerstones of systems biology. This dissertation developed and applied approaches from statistical machine learning for learning condition-specific networks that describe condition-specific behavior of cells at a systems level. Development of these approaches required us to address some general questions in network learning and some questions that are specific to the problem of learning condition-specific networks. In this chapter, we discuss the work described in the previous chapters and relate it to the big picture of condition-specific network learning.

8.1 Discussion

Representing biological networks as undirected graphs One of the first modeling questions for learning condition-specific networks concerns the framework used to represent biological networks. Because we focus on identifying general statistical dependencies, which are correlative rather than causative, undirected probabilistic graphical models provide a natural representation of biological networks. However, structure learning of these models using likelihood score-based methods is difficult because the normalization constant required to generate a valid joint distribution is intractable. To address this problem, we establish an equivalence between the Markov blanket canonical parameterization (MBCP) of Abbeel *et al.* and local per-variable canonical parameters. We then develop a structure learning algorithm based on this equivalence that learns the structure by finding the best consistent Markov blanket of each random variable. Although this idea of learning structure via Markov blanket estimation is well-known in graphical model structure learning, the work described in Chapter 4 makes two important contributions: (a) we derived our algorithm from Abbeel *et al.*'s structure learning algorithm, establishing a connection between MBCP-based structure learning and Markov blanket estimation algorithms, (b) we demonstrated that imposing structural consistency is important for learning high-quality network structures.

Although we discuss the benefits of undirected graphical model representation for biological networks, it is important to empirically justify our model of choice over directed graphical models, which have also been used to represent biological networks. We evaluated the quality of the inferred networks using both directed and undirected graphical models, and empirically demonstrated on simulated datasets with known ground truth, undirected graphs indeed capture the network structure better than directed graphs.

Validation The problem of network learning is essentially unsupervised in nature: we don't know the true answer. This makes validation of the inferred dependencies a difficult problem. We addressed this issue by : (a) developing a simulation framework that generates the network and the data to be used for network inference and subsequent comparison with the true networks, (b) developing metrics for evaluating how well algorithms capture higher-order dependencies – dependencies among three or more random variables. Although the use of a simulation framework to validate unsupervised learning algorithms is a common technique in machine learning, developing a simulation framework that realistically models biological networks is a non-trivial challenge. Our differential-equation based regulatory network simulator constructs networks that are more biologically plausible than existing simulators by: (a) accounting for protein expression levels and (b) formulating kinetic models that capture the combinatorial control among the transcription factors.

While there is no doubt that biological networks have higher-order dependencies [101, 110, 75], it was not clear if this affects the performance of different algorithms, and, therefore the choice of models. Using our validation framework we measured performance of different algorithms as a function of the extent of higher-order dependencies in the network structure. We compared algorithms capturing higher-order dependencies as well as those that approximated higher-order dependencies via pairwise dependencies. Our results suggested that high-indegree and size of transcription factor (TF) complexes are characteristics of biological networks that influence algorithm performance. In particular, algorithms capturing higher-order dependencies performed well on networks with high proportions of nodes with high indegree, or with many large TF complexes, whereas algorithms capturing pair-wise dependencies performed well on networks consisting mostly of nodes with low indegree ($= 1$).

Hidden versus observed condition variables The condition variables are global variables that affect the wiring of the underlying network. Because the values of the condition variables may not be known *a priori*, general models that can infer the condition variables are required. We developed several approaches based on mixture models that can automatically infer the condition value for each data point, but we found that the generality of these models comes at a cost. In particular, if the datasets from the two conditions are highly similar, a model that is given the condition variable is able to perform much better than a model that must infer the condition variable. However, knowing the condition variable is not enough, as evidenced in the independent learner (INDEP) results, which did not capture the shared information as well as NIPD and some of the regulatory processes identified by both NIPD and the approaches based on mixture-model (MUG, PVEM, GC-COND). Because the mixture model framework automatically shares the data points among the component models, it captures the shared parts of the networks as well or better than INDEP.

Different granularities of condition-specificity Condition-specific behavior is complex, which requires networks driving this behavior to be modular and to be composed of reusable parts [75]. It is likely that certain parts of the network are universally required in any kind of condition, and certain parts are unique to a condition. For example, if conditions were stresses, components of the network involved in post-translational modification may be shared across a large set of conditions, whereas respiration related processes may be triggered only under aerobic conditions. This shared response pattern is further supported by the presence of target genes under the control of single master regulators. If conditions are species, the shared parts could correspond to evolutionarily conserved core processes, and the unique parts to the processes necessary for accomplishing species-specific functions. To mathematically model the different types of sharing, we formulated different types of models in Chapter 6, where a condition variable was allowed to influence different proportions of the network. Note that this flexibility is not available in models that assume the condition variable is observed, because the condition variable is specified for the entire set of measurements per microarray (data point) and not for subsets of genes.

A surprising result from our comparative analysis of different algorithms was that even if the generative and target models come from the same family of distributions, the target model may not necessarily perform the best. In particular, there are characteristics of the underlying union network such as the connectivity of the graph, that also influence algorithm performance. We found that, on simulated data, the Mixture of Graphs model (MUG), in which a single condition variable influences the entire network, performed well on networks with a large giant component, irrespective of the generative model. In contrast, the Per-variable EM (PVEM) model, where each network node has its own condition variable, did not perform as well, even though the simulated data was generated from a model that closely obeyed the assumptions of PVEM. The poor behavior of the PVEM model was again due to the overly complex model that did not obey the constraints of the condition assignments. The MUG model performed so well because by having a single condition variable, it naturally obeyed the constraints of the condition variable. In contrast, when the underlying union network had several connected components, PVEM and the graph component models were able to capture the sharing patterns much better than the MUG model.

In addition to this observation of how the topology and the generative model together influence algorithm performance, this work also produced some contributions to the field of machine learning. In particular, we developed and implemented structure learning algorithms for each of these models, which are all based on some form of mixture modeling on graphs. Prior to this work, the only mixture model over graphs is the mixture of directed graphs. Our PVEM model is similar in nature to models describing interventional distributions [38]. However, the intervention variable selects different parameter sets, whereas we select different structures, and, therefore also parameters. The NIPD model, too, is an extension to the existing multi-net framework, where we incorporate explicit data pooling between the conditions.

Information sharing across conditions Component re-use is one of the principles of software design, and the same is true for living systems. One of the questions we asked in this dissertation was how to capture and exploit the shared information that arises from component re-use across conditions. As discussed above, we developed several models that can capture shared information either by explicitly pooling data from non-singleton sets, by using mixture models (which allows data points to be shared across the networks), or by more fine-grained sharing via parameter tying.

Regardless of the condition variable being observed or not, we found that a model that did perform sharing was beneficial. Approaches such as INDEP that learn networks for each condition independently and then compare the resulting networks, are more likely to learn different networks, making it difficult to identify the similarities across conditions. Application of INDEP and NIPD to microarray data from two yeast populations showed that many of subgraphs that would be considered specific to each population by INDEP were actually shared biological processes that must be activated in both populations, irrespective of their morphological and physiological differences. Application to species-specific networks also demonstrated the potential benefits of models exploiting shared information. This suggests that models that explicitly share information are beneficial for capturing the shared parts of the networks and to learn better condition-specific networks.

Biological implications of this work Analysis of condition-specific response is a central theme of molecular biology. In this dissertation, we provided a general framework for viewing condition-specific response as a network and the condition as a global variable that triggers changes in the wiring of the cellular networks. We demonstrated the feasibility of this framework on two examples of the condition variable: one representing different cell populations and the other representing different species. In both, we identified biologically meaningful dependencies including some that are consistent with existing knowledge, and some novel findings.

In particular, in yeast stationary phase we found respiration-related processes to be conserved in both populations, and we identified several regulatory and metabolic processes that capture key physiological characteristics of the quiescent and non-quiescent populations. We also identified candidate triplets that can be experimentally tested using double gene knock-outs, and that contribute to our understanding of differentiated cell populations in yeast stationary phase. In species-specific networks, we found several metabolic processes (pyruvate metabolism, fatty acid biosynthesis) to be conserved in yeast and fly. The implication of this is that these processes could be studied in yeast, a model organism with the widest collection of genome-scale reagents, and the experimental findings can be transferred from the yeast model system to the fly model system.

The results from our application to these diverse definitions of what a condition means, has hopefully convinced the reader of the generality of our framework and of condition-specific network learning as such. Different tissues and diseases represent yet more examples of different conditions, and learning networks for such conditions will provide insight into developmental body plan and insights into the mechanisms and causes of different diseases.

8.2 Future work

This dissertation establishes ground-work for important future enhancements that will allow us to systematically identify the parts, and the wiring among them, that determine stage-specific, tissue-specific, and disease specific behavior in whole organisms. We expand on them below:

Taking condition-specificity to tissues, diseases and metagenomes In this dissertation, we demonstrated our approach on conditions representing cell populations and species. However, our general approach to condition-specific network learning applies to a wider class of problems in biology. One application of our condition-specific network learning approaches is to consider tissues as different conditions and learn tissue-specific networks [37], which can provide insights into cell-fate determination and development in higher organisms.

Our approaches can also be applied to infer disease-specific networks from genome-wide measurements of human transcripts, which can identify differences and similarities among diseases and help diagnosis of new types of disease [30, 26]. On a similar note, our methods apply to genome-wide association studies, which are rapidly growing thanks to next generation sequencing [96]. These studies assay genotypic variation from populations of people to identify genetic markers of disease susceptibility. Learning networks from these data, treating each population type as a condition, can identify higher-order relationships among the markers that likely influence disease susceptibility.

Finally, our approaches are applicable to metagenomic data [54, 57], which capture metabolic or transcriptomic profiles of different eco-systems of multiple species, most of which cannot be cultured or sequenced in the lab. We can treat each species as a random variable in our networks, and different eco-systems as condition variables. The inferred dependencies among the species can then be used to obtain a better understanding of novel species and eco-systems as a whole.

Inferring causal, condition-specific networks Fixing a broken system requires us to find the *cause* of the failure. Similarly, to understand life we need to understand what causes cells to transition between different disease and healthy states. In this work, we have developed approaches for obtaining probabilistic network descriptions of the average state of the cell. The next step is to specify the causes and the effects in the network. However, traditional methods of identifying causal interactions require many expensive perturbations and assume *no hidden variables*, which is unrealistic for real biological systems. Fortunately, recently available expression QTL data can be used to infer causal interactions under the assumption that changes in gene expression are caused by changes in the DNA sequence [112]. Integration of these type of genotypic and phenotypic data within the framework of condition-specific network learning will allow us to identify the causal mechanisms governing the condition-specific responses.

Computational complexity of exhaustive enumeration of condition sets Several of our models for condition-specific network learning exhaustively enumerate over all subsets of the condition sets. This poses serious computational challenges for moderately sized (dozen) condition sets. This is compounded with the statistical challenges of data sparsity arising from learning exponentially (in the size of the condition set) many networks. Although such datasets are not available as of the present, with next generation sequencing, datasets assaying dynamics of gene expression over time per tissue or developmental stage are a near possibility. This requires us to devise both fast search algorithms for network structure and heuristics that enable us to avoid exhaustive enumeration of the condition sets.

Relaxing the topologies of the inferred networks The undirected graphical models that are inferred using our algorithms constraint the size of the Markov blanket to a fixed k. While this allows us to tractably learn the network structure, it tends to produce networks topologies that are regular. A possible extension to our work is to incorporate a per-variable regularization term that uses a more flexible regularization of each random variable [121]. The per-variable regularization term can be used in an automatic MDL setting or can be based on biological prior knowledge of genes that are likely to have high degrees e.g. transcription factors. This extension would allow us to more easily capture skewed distributions of many biological networks.

A more general hierarchical Bayesian framework for parameter sharing Our parameter tying approach to sharing data across conditions is based on a simplistic idea of constraining weights and applies only for the conditional Gaussian case. However, the general idea of information sharing across conditions is well-known in the multi-task learning field of machine learning [106]. These approaches share data across conditions by using a hierarchical Bayesian framework [53], where a global hyper-prior is used to share information across parameter vectors for each component of the mixture model. This provides a more principled framework to parameter sharing and has been recently extended to a non-parametric setting via Gaussian processes [145, 144, 82]. However, most of multi-task learning approaches solve related regression problems, rather than solving the more difficult problem of learning network structures. Further, in this framework, the task (or condition) label is assumed to be known for each datapoint. Extending our condition-specific network learning algorithms to a hierarchical Bayesian framework is a fruitful direction of future research, which can make contributions to machine learning and also address a wider range of problems in biology.

Generative model of edges rather than expression variables The work done in this dissertation is based on probabilistic generative models of gene expression. In this formulation, a random variable models the expression value of each gene. An alternate formulation to the problem is to treat the edges themselves as random variables and develop generative mixture models for the edges. In particular, we want a mixture model over network edges, similar to a Latent Dirichlet Allocation (LDA) model for text documents [17]. To draw an analogy, each word corresponds to an edge, a document corresponds to a network and the topics correspond to the hidden conditions. An example of this idea was implemented by Aukia *et al.* for a large social network describing music selections [7]. However, this approach assumes that the network structure is known and the only inference problem is inferring the component assignment for each edge. Implementing this kind of a generative model in our target problems, where the network structure is not known, will require novel extensions to both LDA models and network structure learning.

Our understanding of the biology of the cell is advancing from a single gene to the entire genome, and functional networks have been key in obtaining this systemic view of the cell. Inference and analysis of condition-specific networks provides yet another stepping stone in our quest for knowledge about the inner-workings of a living system.

Appendices

Appendix A

Equivalence of Markov blanket and per-variable canonical parameters

In this appendix we give some background information of Markov random fields (MRFs). We describe the Hammersley Clifford theorem, the original canonical parameterization and Markov blanket canonical parameterization (MBCP), and then give the proof of equivalence between the MBCP and the per-variable canonical parameters.

A.1 Hammersly-Clifford theorem and canonical potentials

The Hammersly-Clifford theorem establishes a one-to-one relationship between MRFs and strictly positive distributions such as the Gibbs distributions. The *canonical potentials* (also called $\mathcal{N}$-potentials [107]) are used together with the Möbius inversion theorem to prove the Hammersly-Clifford theorem [80]. The canonical potential for a subset $\mathbf{D} \subseteq \mathbf{X}$

is defined using a *default joint instantiation*, $\overline{\mathbf{x}} = \{\overline{x}_1, \cdots, \overline{x}_{|\mathbf{X}|}\}$ to $\mathbf{X}$ as:

$$\psi^*_{\mathbf{D}}(\mathbf{D} = \mathbf{d}) = \\ \exp\left(\sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D} \backslash \mathbf{U}|} \log P(\mathbf{X} = \sigma(\mathbf{U}, \mathbf{X}, \mathbf{d}))\right)$$

where $\sigma(\mathbf{A}, \mathbf{B}, \mathbf{c})$ is an assignment function to variables $X_k \in \mathbf{B}$ such that $\sigma(\mathbf{A}, \mathbf{B}, \mathbf{c})[k] = c_k$, if $X_k \in \mathbf{A}$ and $\sigma(\mathbf{A}, \mathbf{B}, \mathbf{c})[k] = \overline{x}_k$ if $X_k \notin \mathbf{A}$. σ returns an assignment for all variables in $\mathbf{B}$.

The Mobius inversion states that for any real functions f and g over subsets $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$

$$f(\mathbf{A}) = \sum_{\mathbf{B} \subseteq \mathbf{A}} g(\mathbf{B}), \qquad \text{is true iff,} \\ g(\mathbf{B}) = \sum_{\mathbf{C} \subseteq \mathbf{B}} (-1)^{|\mathbf{B} \backslash \mathbf{C}|} f(\mathbf{C}) \tag{A.1}$$

The joint probability distribution for a MRF using canonical potentials is defined to be: $P(\mathbf{X} = \mathbf{x}) = P(\overline{\mathbf{x}}) \prod_{\mathbf{D} \in \mathcal{C}} \psi^*_{\mathbf{D}}$, where $\mathcal{C}$ is the set of maximal cliques in the graph. This is true by an application of the Möbius inversion and setting $\psi^*_{\mathbf{D}} = 0$ for all $\mathbf{D} \notin \mathcal{C}$ [80, 107].

A.2 Markov blanket canonical parameterization

The computation of the canonical potentials is not feasible for real-world domains as they require the estimation of the full joint distribution [1]. Markov Blanket canonical parameterization, developed by Abbeel *et al.*, allows the computation of global canonical potentials over $\mathbf{X}$, using local conditional functions called *Markov blanket canonical factors* (MBCFs).

The MBCF, $\widetilde{\psi}$ for a set $\mathbf{D} \subseteq \mathbf{X}$ is estimated using $\mathbf{D}$ and its Markov blanket (MB). The MB, $\mathbf{M}_i$ of a variable X_i, is the set of immediate neighbors of X_i in $\mathcal{G}$ and renders

X_i conditionally independent of other variables, i.e., $P(X_i|\mathbf{X} \setminus \{X_i\}) = P(X_i|\mathbf{M}_i)$. The MB, $\mathbf{M_D}$ of a set $\mathbf{D}$, is $\left(\bigcup_j \mathbf{M}_j\right) \setminus \mathbf{D}$ for all $X_j \in \mathbf{D}$. The MBCF, $\widetilde{\psi}$ for $\mathbf{D}$ is also defined using the default joint instantiation, $\overline{\mathbf{x}} = \{\overline{x}_1, \cdots, \overline{x}_{|\mathbf{X}|}\}$ as:

$$\widetilde{\psi}_{\mathbf{D}}(\mathbf{D}=\mathbf{d}) = \exp\Bigg(\sum_{\mathbf{U}\subseteq\mathbf{D}} (-1)^{|\mathbf{D}\setminus\mathbf{U}|}\log P(\mathbf{D}=\sigma(\mathbf{U},\mathbf{D},\mathbf{d})| \\ \mathbf{M_D}=\sigma(\mathbf{U},\mathbf{M_D},\mathbf{d}))\Bigg), \tag{A.2}$$

For MRFs of unknown structure, MBCFs are identified by searching exhaustively among all subsets $\mathbf{F}_i \subset \mathbf{X}$, up to size l and finding MBs for each $\mathbf{F}_i$. Unfortunately, exhaustive enumeration of variable subsets becomes impractical for moderately sized networks [1]. We show that the MBCFs can be further reduced to smaller per-variable canonical factors, which are computed using an RV and its Markov blanket.

A.2.1 Per-variable MB canonical factors

We now show that the MBCFs can be replaced by smaller, local functions: *per-variable MB canonical factors*, which does not require enumeration of all subsets up to size l. Specifically, for every MB canonical factor $\widetilde{\psi}$ there exists an equivalent per-variable canonical factor ψ^+. To illustrate how the per-variable factors are derived from MBCFs, we first consider a specific case of $\mathbf{D} = \{X_i, X_j\}$ in Eq A.2 (Section A.2.1), followed by a proof for the general case (Section A.2.1).

Special case of two variables

Let $\mathbf{D} = \{X_i, X_j\}$ and $\mathbf{d} = \{x_i, x_j\}$. Note, because $\mathbf{D} \cap \mathbf{M_D} = \emptyset$, $\sigma(\mathbf{U}, \mathbf{M_D}, \mathbf{d}) = \overline{\mathbf{m}}_{\mathbf{d}}$, the default instantiation to $\mathbf{M_D}$ from $\overline{\mathbf{x}}$. We first expand the sum inside the exponential of

Eq A.2 with $\mathbf{D} = \{X_i, X_j\}$:

$$\begin{aligned}
&\sum_{\mathbf{U}\subseteq\mathbf{D}} (-1)^{|\mathbf{D}\setminus\mathbf{U}|}\log P(\{X_i, X_j\} = \\
&\qquad \sigma(\mathbf{U}, \{X_i, X_j\}, \mathbf{d})|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&= (-1)^{|\{X_i,X_j\}|}\log P(X_i = \overline{x}_i, X_j = \overline{x}_j|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\quad +(-1)^{|\{X_j\}|}\log P(X_i = x_i, X_j = \overline{x}_j|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\quad (-1)^{|\{X_i\}|}\log P(X_i = \overline{x}_i, X_j = x_j|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\quad +(-1)^{|\emptyset|}\log P(X_i = x_i, X_j = x_j|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}})
\end{aligned} \tag{A.3}$$

where the first term corresponds to $U = \emptyset$, the second term corresponds to $U = \{X_i\}$ and so on. Applying the chain rule to every term in the RHS:

$$\begin{aligned}
&= \log[P(X_i = \overline{x}_i|X_j = \overline{x}_j, \mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\qquad P(X_j = \overline{x}_j|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}})] \\
&\quad -\log[P(X_i = x_i|X_j = \overline{x}_j, \mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\qquad P(X_j = \overline{x}_j|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}})] \\
&\quad -\log[P(X_i = \overline{x}_i|X_j = x_j, \mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\qquad P(X_j = x_j|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}})] \\
&\quad +\log[P(X_i = x_i|X_j = x_j, \mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\qquad P(X_j = x_j|\mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}})]
\end{aligned}$$

We find that all $\log P(X_j|\mathbf{M}_{\mathbf{D}})$ terms cancel producing:

$$\begin{aligned}
&= \log P(X_i = \overline{x}_i, |X_j = \overline{x}_j, \mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\quad -\log P(X_i = x_i|X_j = \overline{x}_j, \mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\quad -\log P(X_i = \overline{x}_i|X_j = x_j, \mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}}) \\
&\quad +\log P(X_i = x_i|X_j = x_j, \mathbf{M}_{\mathbf{D}} = \overline{\mathbf{m}}_{\mathbf{d}})
\end{aligned} \tag{A.4}$$

This allows $\widetilde{\psi}$ to be rewritten as:

$$\widetilde{\psi}_{\mathbf{D}}(\mathbf{D}=\mathbf{d}) = \exp\Bigg(\sum_{\mathbf{U}\subseteq\mathbf{D}} (-1)^{|\mathbf{D}\backslash\mathbf{U}|}\log P(X_i = \sigma(\mathbf{U},\{X_i\},\mathbf{d})|\{X_j\}\cup\mathbf{M_D} = \sigma(\mathbf{U},\{X_j\}\cup\mathbf{M_D},\mathbf{d}))\Bigg) \tag{A.5}$$

We assert further independence in Eq A.5 because X_i is independent of all variables other than $\mathbf{M}_i$. This allows us to write the original MBCF for $\{X_i, X_j\}$ as the per-variable canonical factor:

$$\psi^{+}_{\mathbf{D}}(\mathbf{D}=\mathbf{d}) = \exp\Bigg(\sum_{\mathbf{U}\subseteq\mathbf{D}} (-1)^{|\mathbf{D}\backslash\mathbf{U}|}\log P(X_i = \sigma(\mathbf{U},\{X_i\},\mathbf{d})|\mathbf{M}_i = \sigma(\mathbf{U},\mathbf{M}_i,\mathbf{d}))\Bigg) \tag{A.6}$$

Thus we have equivalent the per-variable canonical factor, ψ^{+} from the original MBCF $\widetilde{\psi}$ in Eq A.2.

General case

We now state the equivalence between per-variable and MB canonical factors more formally:

Theorem A.2.1 *Every MBCP, $\widetilde{\psi}_{\mathbf{D}}$, of the form in Eq A.2 possesses an equivalent per-variable factor, $\psi^{+}_{\mathbf{D}}(\mathbf{D}=\mathbf{d}) = exp\Big(\sum_{\mathbf{U}\subseteq\mathbf{D}}(-1)^{|\mathbf{D}\backslash\mathbf{U}|}log P(X_i = \sigma(\mathbf{U},\{X_i\},\mathbf{d})|\mathbf{M}_i = \sigma(\mathbf{U},\mathbf{M}_i,\mathbf{d}))\Big)$, where $X_i \in \mathbf{D}$.*

Proof The proof of this equivalence involves two steps: (a) deriving ψ^{+} from $\widetilde{\psi}$ for any general $\mathbf{D}$, and (b) identifying neighbors of an RV and making independence assertions described by the graph structure.

To prove (a) we select an arbitrary $X_i \in \mathbf{D}$. We replace each log$P(\mathbf{D}|\mathbf{M_D})$ in $\widetilde{\psi}$ by log$(P(X_i|\mathbf{D} \setminus \{X_i\} \cup \mathbf{M_D})P(\mathbf{D} \setminus \{X_i\}|\mathbf{M_D}))$. We have $2^{|\mathbf{D}|}$ number of log$P(\mathbf{D} \setminus \{X_i\}|\mathbf{M_D})$ terms, one for each $\mathbf{U} \subseteq \mathbf{D}$. X_i does not occur in these terms, as we have conditioned on it. These terms can be grouped into two sets, $\mathcal{S}_{od}$ and $\mathcal{S}_{ev}$, where $\mathcal{S}_{od}$ and $\mathcal{S}_{ev}$ correspond to subsets of $\mathbf{D}$ with odd and even number of elements, respectively. Assuming $|\mathbf{D}|$ is even, all elements in $\mathcal{S}_{od}$ have a $-$ve sign and all elements in S_{ev} have a $+$ve sign. Further, for every $t \in \mathcal{S}_{ev}$ corresponding to $\mathbf{U} \subseteq \mathbf{D}$ there exists $t' \in \mathcal{S}_{od}$ corresponding to $\mathbf{U}' \subseteq \mathbf{D}$, such that $\mathbf{U}$ and $\mathbf{U}'$ differ only in X_i. Because X_i does not occur in either t or t', these two terms cancel. Applying this to all elements of $\mathcal{S}_{od}$ and $\mathcal{S}_{ev}$, the two subsets cancel each other, thus proving (a). If $|\mathbf{D}|$ is odd, elements of $\mathcal{S}_{od}$ and $\mathcal{S}_{ev}$ have $+$ve and $-$ve signs, respectively, and the rest of the argument follows.

The final step is to identify the neighbors of X_i and using the local Markov property, $P(X_i|\mathbf{D} \setminus \{X_i\} \cup \mathbf{M_D}) = P(X_i|\mathbf{M}_i)$, for strictly positive distributions [80] □.

The equivalence of the per-variable factors and MBCFs implies that, instead of searching over all size l subsets of $\mathbf{X}$, we can estimate canonical factors by searching for MBs of individual RVs. Assuming that the MBs are estimated correctly, Eq A.6 will produce the same canonical factors as Eq A.2. Our structure learning algorithm therefore requires the estimation of MBs of each RV. We only need to ensure structural consistency. Searching for n MBs, as opposed to n^l MBs in MBCF, saves us $O(n^{l-1})$ computations.

The per-variable canonical factors and MBCFs do not deny the hardness of computing likelihood in MRFs [1]. This is because computing $P(\mathbf{X} = \overline{\mathbf{x}})$ is equivalent to computing $\frac{1}{Z}$.

Appendix B

Deriving the normalization term and quantifying the correction

Our conditional distribution for each variable X_i in condition c is

$$P(X_i = x_{cid} | \mathbf{M}_{ci} = \mathbf{m_{cid}}) \propto \prod_{\mathbf{E} \in \text{powerset}(\mathcal{C}) \,:\, c \in \mathbf{E}} P(X_i | \mathbf{M}^*_{\mathbf{E}i}),$$

Consider the case where we have two conditions $\mathcal{C} = \{1, 2\}$. The conditional for condition 1 is

$$P(X_i = x_{1id} | \mathbf{M}_{1i} = \mathbf{m_{1id}}) \propto P(X_i = x_{1id} | \mathbf{M}^*_{1i}) P(X_i = x_{1id} | \mathbf{M}^*_{3i})$$

where 1 and 3 denote the condition sets $\{1\}$ and $\{1, 2\}$. Assuming conditional Gaussians for each condition set

$$P(X_i = x_{1id} | \mathbf{M}^*_{1i} = \mathbf{m}^*_{1id}) = \frac{1}{\sqrt{2\pi\sigma^2_{1i}}} \text{exp}\left(\frac{(x_{1id} - \mathbf{w}_1^\mathsf{T} \mathbf{m}^*_{1id})^2}{-2\sigma^2_{1i}}\right)$$

We let μ_{1id} denote $\mathbf{w}_1^\mathsf{T} \mathbf{m}_{1id}$, that is the mean for the conditional Gaussian. The product of conditionals in 1 and 3 is

$$\frac{1}{2\pi\sigma_{1i}\sigma_{3i}} \text{exp}\left(\frac{(x_{1id} - \mu_{1id})^2}{-2\sigma^2_{1i}} + \frac{(x_{1id} - \mu_{3id})^2}{-2\sigma^2_{3i}}\right)$$

$$= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\left(-\frac{1}{2}\left[\frac{(x_{1id}-\mu_{1id})^2\sigma_{3i}^2+(x_{1id}-\mu_{3id})^2\sigma_{1i}^2}{\sigma_{1i}^2\sigma_{3i}^2}\right]\right)$$

$$= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\left(-\frac{1}{2}\left[\frac{(x_{1id}^2+\mu_{1id}^2-2x_{1id}\mu_{1id})\sigma_{3i}^2+(x_{1id}^2+\mu_{3id}^2-2x_{1id}\mu_{3id})\sigma_{1i}^2}{\sigma_{1i}^2\sigma_{3i}^2}\right]\right)$$

$$\begin{aligned}= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{1}{2\sigma_{1i}^2\sigma_{3i}^2}&[(x_{1id}^2+\mu_{1id}^2-2x_{1id}\mu_{1id})\sigma_{3i}^2\\ &+(x_{1id}^2+\mu_{3id}^2-2x_{1id}\mu_{3id})\sigma_{1i}^2]\Big)\end{aligned}$$

$$\begin{aligned}= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{1}{2\sigma_{1i}^2\sigma_{3i}^2}[(x_{1id}^2(\sigma_{3i}^2+\sigma_{1i}^2)-2x_{1id}(\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2)\\ +\mu_{1id}^2\sigma_{3i}^2+\mu_{3id}^2\sigma_{1i}^2]\Big)\end{aligned}$$

$$\begin{aligned}= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{\sigma_{3i}^2+\sigma_{1i}^2}{2\sigma_{1i}^2\sigma_{3i}^2}[(x_{1id}^2-\frac{2x_{1id}(\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2)}{\sigma_{3i}^2+\sigma_{1i}^2}\\ +\frac{\mu_{1id}^2\sigma_{3i}^2+\mu_{3id}^2\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}]\Big)\end{aligned}$$

Completing the square

$$\begin{aligned}\frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{\sigma_{3i}^2+\sigma_{1i}^2}{2\sigma_{1i}^2\sigma_{3i}^2}\Big[(x_{1id}^2-\frac{2x_{1id}(\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2)}{\sigma_{3i}^2+\sigma_{1i}^2}+\frac{(\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2)^2}{(\sigma_{3i}^2+\sigma_{1i}^2)^2}\\ -\frac{(\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2)^2}{(\sigma_{3i}^2+\sigma_{1i}^2)^2}+\frac{\mu_{1id}^2\sigma_{3i}^2+\mu_{3id}^2\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}\Big]\Big)\end{aligned}$$

$$\begin{aligned}\frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{1}{2\frac{\sigma_{1i}^2\sigma_{3i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}}\Big[\Big(x_{1id}-\frac{\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}\Big)^2-\frac{(\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2)^2}{(\sigma_{3i}^2+\sigma_{1i}^2)^2}\\ +\frac{\mu_{1id}^2\sigma_{3i}^2+\mu_{3id}^2\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}\Big]\Big)\end{aligned}$$

$$= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{1}{2\frac{\sigma_{1i}^2\sigma_{3i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}}\Big(x_{1id}-\frac{\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}\Big)^2\Big)$$
$$\exp\Big(-\frac{\sigma_{3i}^2+\sigma_{1i}^2}{2(\sigma_{1i}^2\sigma_{3i}^2)}[\frac{-(\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2)^2+(\mu_{1id}^2\sigma_{3i}^2+\mu_{3id}^2\sigma_{1i}^2)(\sigma_{3i}^2+\sigma_{1i}^2)}{(\sigma_{3i}^2+\sigma_{1i}^2)^2}]\Big)$$

$$= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{1}{2\frac{\sigma_{1i}^2\sigma_{3i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}}\Big(x_{1id}-\frac{\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}\Big)^2\Big)$$
$$\exp\Big(-\frac{1}{2\sigma_{1i}^2\sigma_{3i}^2}[\frac{-\mu_{1id}^2\sigma_{3i}^4-\mu_{3id}^2\sigma_{1i}^4-2\mu_{1id}\mu_{3id}\sigma_{3i}^2\sigma_{1i}^2+(\mu_{1id}^2\sigma_{3i}^2+\mu_{3id}^2\sigma_{1i}^2)(\sigma_{3i}^2+\sigma_{1i}^2)}{\sigma_{3i}^2+\sigma_{1i}^2}]\Big)$$

$$= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{1}{2\frac{\sigma_{1i}^2\sigma_{3i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}}\Big(x_{1id}-\frac{\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}\Big)^2\Big)$$
$$\exp\Big(-\frac{1}{2\sigma_{1i}^2\sigma_{3i}^2}[\frac{-2\mu_{1id}\mu_{3id}\sigma_{3i}^2\sigma_{1i}^2+\mu_{3id}^2\sigma_{1i}^2\sigma_{3i}^2+\mu_{1id}^2\sigma_{3i}^2\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}]\Big)$$

$$= \frac{1}{2\pi\sigma_{1i}\sigma_{3i}}\exp\Big(-\frac{1}{2\frac{\sigma_{1i}^2\sigma_{3i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}}\Big(x_{1id}-\frac{\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}\Big)^2\Big)$$
$$\exp\Big(-\frac{1}{2}[\frac{-2\mu_{1id}\mu_{3id}+\mu_{3id}^2+\mu_{1id}^2}{\sigma_{3i}^2+\sigma_{1i}^2}]\Big)$$

$$= \frac{1}{\sqrt{2\pi}\frac{\sigma_{1i}\sigma_{3i}}{\sqrt{\sigma_{1i}^2+\sigma_{3i}^2}}}\exp\Big(-\frac{1}{2\frac{\sigma_{1i}^2\sigma_{3i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}}\Big(x_{1id}-\frac{\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2}\Big)^2\Big)$$
$$\frac{1}{\sqrt{2\pi(\sigma_{1i}^2+\sigma_{3i}^2)}}\exp\Big(-\frac{1}{2}[\frac{-2\mu_{1id}\mu_{3id}+\mu_{3id}^2+\mu_{1id}^2}{\sigma_{3i}^2+\sigma_{1i}^2}]\Big)$$

Thus we can think of $P(X_i|\mathbf{M}_{1i})P(X_i|\mathbf{M}_{3i})$ as a product

$$\mathcal{N}(x_{1id}|\frac{\mu_{1id}\sigma_{3i}^2+\mu_{3id}\sigma_{1i}^2}{\sigma_{3i}^2+\sigma_{1i}^2};\frac{\sigma_{1i}\sigma_{3i}}{\sqrt{\sigma_{1i}^2+\sigma_{3i}^2}})\mathcal{N}(\mu_{1id}|\mu_{3id};\sqrt{\sigma_{1i}^2+\sigma_{3i}^2})$$

The normalization constant is therefore $\mathcal{N}(\mu_{1id}|\mu_{3id};\sqrt{\sigma_{1i}^2+\sigma_{3i}^2})$. The valid probability distribution is given by the first Gaussian and this is what on which we must compute the pseudo likelihood. However, we work directly with the product of conditionals and this has two ramifications: (a) the score improvement can be computed very efficiently, (b) the second Gaussian acts as a smoothing term over the parameter μ_{1id}. In particular, if we were estimating a new parameter μ_{1id}, the second Gaussian specifies the probability of the new μ_{1id} using a Gaussian centered around the mean computed from the pooled dataset. Finally the change in score improvement due to this term is a ratio of Gaussians and does not affect the overall score of our model significantly. In our experiments, we find this score to have better performance than if we were to subtract out this term.

B.1 Correction for the unnormalized score

Because the MB per condition set independently influence the conditional, the pseudo likelihood PLLV$(X_i, \mathbf{M}_{ci}, c)$ decomposes as $\sum_{\mathbf{E}\text{ s.t. }c\in\mathbf{E}} PLLV(X_i, \mathbf{M}^*_{\mathbf{E}i}, c)$. The net score improvement of adding an edge $\{X_i, X_j\}$ to a condition set $\mathbf{C}$ as given by:

$$\Delta\text{Score}_{\{X_i,X_j\},\mathbf{C}} = \sum_{c\in\mathbf{C}}\sum_{d=1}^{|\mathcal{D}_c|}\text{PLLV}(X_i,\mathbf{M}_{ci}\cup\{X_j\},c)-\text{PLLV}(X_i,\mathbf{M}_{ci},c)+ \\ \text{PLLV}(X_j,\mathbf{M}_{cj}\cup\{X_i\},c)-\text{PLLV}(X_j,\mathbf{M}_{cj},c) \tag{B.1}$$

Considering only the effect on X_i. Let $\mathcal{C}=1,2$, $\mathbf{C}=\{1\}$.

$$\Delta\text{Score}_{\{X_i,X_j\},\mathbf{1}} = \text{PLLV}(X_i,\mathbf{M}_{1i}\cup\{X_j\},1)-\text{PLLV}(X_i,\mathbf{M}_{1i},1) \tag{B.2}$$

$$P(X_i = x_{1id}|\mathbf{M}_{1i} = \mathbf{m_{1id}}) = \frac{1}{\mathcal{N}(\mu_{1id}|\mu_{3id}, \sqrt{\sigma_1^2 + \sigma_3^2})} \prod_{\mathbf{E} \in \text{powerset}(\mathcal{C})\,:\, c \in \mathbf{E}} P(X_i|\mathbf{M}^*_{\mathbf{E}i}),$$

where $\mu_{1id} = \mathbf{w}_{1i}^{\mathsf{T}}\mathbf{m}_{1id}$ and $\mu_{3id} = \mathbf{w}_{3i}^{\mathsf{T}}\mathbf{m}_{3id}$. Here μ_{3id} and σ_{3i} are the parameters of the conditional Gaussian from condition set $\{1, 2\}$. The pseudo likelihood of $\mathcal{D}_1$ is

$$PLL = \sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}_{1i} = \mathbf{m_{1id}}). \tag{B.3}$$

which for the conditional Gaussian model is

$$\begin{aligned} PLL = \sum_{d=1}^{|\mathcal{D}_1|} \log \frac{1}{\mathcal{N}(\mu_{1id}|\mu_{3id}, \sqrt{\sigma_{1i}^2 + \sigma_{3i}^2})} \\ P(X_i = x_{1id}|\mathbf{M}^*_{1i} = \mathbf{m}^*_{1id}) P(X_i = x_{1id}|\mathbf{M}^*_{3i} = \mathbf{m}^*_{3id}). \end{aligned} \tag{B.4}$$

$$\begin{aligned} PLL_{old} = -\sum_{d=1}^{|\mathcal{D}_1|} \log \mathcal{N}(\mu_{1id}|\mu_{3id}, \sqrt{\sigma_{1i}^2 + \sigma_{3i}^2}) \\ + \sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}^*_{1i} = \mathbf{m}^*_{1id}) \\ + \sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}^*_{3i} = \mathbf{m}^*_{3id}). \end{aligned} \tag{B.5}$$

Now lets look at the score improvement on adding X_j to X_i's Markov blanket in condition 1. This changes $\mathbf{M}^*_{1i}$ to $\mathbf{M}^*_{1i} \cup \{X_j\}$. Let σ^+_{1i} and also $\mathbf{w}^+_{1i}$ be the new parameters of the conditional for X_i on adding X_j to $\mathbf{M}^*_{1i}$. The new pseudo likelihood is

$$\begin{aligned} PLL_{new} = -\sum_{d=1}^{|\mathcal{D}_1|} \log \mathcal{N}(\mu^+_{1id}|\mu_{3id}, \sqrt{\sigma^{2+}_{1i} + \sigma^2_{3i}}) \\ + \sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}^*_{1i} \cup \{X_i\} = \mathbf{m}^*_{1id} \cup \{x_{1jd}\}) \\ + \sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}^*_{3i} = \mathbf{m}^*_{3id}). \end{aligned} \quad (B.6)$$

The score improvement $PLL_{new} - PLL_{old}$ is given by

$$\begin{aligned} \Delta PLL = -\sum_{d=1}^{|\mathcal{D}_1|} \log \mathcal{N}(\mu^+_{1id}|\mu_{3id}, \sqrt{\sigma^{2+}_{1i} + \sigma^2_{3i}}) \\ + \sum_{d=1}^{|\mathcal{D}_1|} \log \mathcal{N}(\mu_{1id}|\mu_{3id}, \sqrt{\sigma^2_{1i} + \sigma^2_{3i}}) \\ + \sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}^*_{1i} \cup \{X_i\} = \mathbf{m}^*_{1id} \cup \{x_{1jd}\}) \\ - \sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}^*_{1i} = \mathbf{m}^*_{1id}) \end{aligned} \quad (B.7)$$

We already take into account the last two terms. The correction we need to add is

$$Corr = -\sum_{d=1}^{|\mathcal{D}_1|} \log \mathcal{N}(\mu^+_{1id}|\mu_{3id}, \sqrt{\sigma^{2+}_{1i} + \sigma^2_{3i}}) + \sum_{d=1}^{|\mathcal{D}_1|} \log \mathcal{N}(\mu_{1id}|\mu_{3id}, \sqrt{\sigma^2_{1i} + \sigma^2_{3i}}) \quad (B.8)$$

$$Corr = -\sum_{d=1}^{|\mathcal{D}_1|} \log \frac{\mathcal{N}(\mu^+_{1id}|\mu_{3id}, \sqrt{\sigma^{2+}_{1i} + \sigma^2_{3i}})}{\mathcal{N}(\mu_{1id}|\mu_{3id}, \sqrt{\sigma^2_{1i} + \sigma^2_{3i}})} \quad (B.9)$$

Now lets consider the case where we are adding the edge $\{X_i, X_j\}$ to condition set $\{1, 2\}$. In this situation, the new parameters are w^{+}_{3id} and σ^{+}_{3id}. Because this affects both conditions 1 and 2, the score improvement needs to take into account both $\mathcal{D}_1$ and $\mathcal{D}_2$. Let PLL^{1}_{old} be the old pseudo likelihood in condition 1, and PLL^{2}_{old} be the old pseudo likelihood in condition 2. Let PLL^{1}_{old} be the same as in equation B.5. PLL^{2}_{old} is given by a similar equation

$$\begin{aligned} PLL^{2}_{old} = &-\sum_{d=1}^{|\mathcal{D}_1|} \log\mathcal{N}(\mu_{2id}|\mu_{3id}, \sqrt{\sigma^{2}_{2i} + \sigma^{2}_{3i}}) \\ &+\sum_{d=1}^{|\mathcal{D}_2|} \log P(X_i = x_{2id}|\mathbf{M}^{*}_{2i} = \mathbf{m}^{*}_{2id}) \\ &+\sum_{d=1}^{|\mathcal{D}_2|} \log P(X_i = x_{2id}|\mathbf{M}^{*}_{3i} = \mathbf{m}^{*}_{3id}). \end{aligned} \tag{B.10}$$

The new PLL in condition 1 is

$$\begin{aligned} PLL^{1}_{new} = &-\sum_{d=1}^{|\mathcal{D}_1|} \log\mathcal{N}(\mu_{1id}|\mu^{+}_{3id}, \sqrt{\sigma^{2}_{1i} + \sigma^{2+}_{3i}}) \\ &+\sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}^{*}_{1i} = \mathbf{m}^{*}_{1id}) \\ &+\sum_{d=1}^{|\mathcal{D}_1|} \log P(X_i = x_{1id}|\mathbf{M}^{*}_{3i} \cup \{X_j\} = \mathbf{m}^{*}_{3id} \cup \{x_{1jd}\}). \end{aligned} \tag{B.11}$$

The correction term will now includes two addition terms

$$\begin{aligned} Corr = &-\sum_{d=1}^{|\mathcal{D}_1|} \log\mathcal{N}(\mu_{1id}|\mu^{+}_{3id}, \sqrt{\sigma^{2}_{1i} + \sigma^{2+}_{3i}}) + \sum_{d=1}^{|\mathcal{D}_1|} \log\mathcal{N}(\mu_{1id}|\mu_{3id}, \sqrt{\sigma^{2}_{1i} + \sigma^{2}_{3i}}) \\ &-\sum_{d=1}^{|\mathcal{D}_2|} \log\mathcal{N}(\mu_{2id}|\mu^{+}_{3id}, \sqrt{\sigma^{2}_{2i} + \sigma^{2+}_{3i}}) + \sum_{d=1}^{|\mathcal{D}_2|} \log\mathcal{N}(\mu_{2id}|\mu_{3id}, \sqrt{\sigma^{2}_{2i} + \sigma^{2}_{3i}}) \end{aligned} \tag{B.12}$$

which on simplification is

$$\begin{aligned} Corr = &-\sum_{d=1}^{|\mathcal{D}_1|} \log \frac{\mathcal{N}(\mu_{1id}|\mu_{3id}^{+}, \sqrt{\sigma_{1i}^{2} + \sigma_{3i}^{2+}})}{\mathcal{N}(\mu_{1id}|\mu_{3id}, \sqrt{\sigma_{1i}^{2} + \sigma_{3i}^{2}})} \\ &-\sum_{d=1}^{|\mathcal{D}_2|} \log \frac{\mathcal{N}(\mu_{2id}|\mu_{3id}^{+}, \sqrt{\sigma_{2i}^{2} + \sigma_{3i}^{2+}})}{\mathcal{N}(\mu_{2id}|\mu_{3id}, \sqrt{\sigma_{2i}^{2} + \sigma_{3i}^{2}})} \end{aligned} \tag{B.13}$$

Appendix C

Deriving a decomposable pseudo likelihood score in the MUG model

Here we begin with the expected pseudo log likelihood and derive a score that decomposes over the individual random variables. Recall from Chapter 6, Section 6.2.3, the expected pseudo likelihood of the completed data is as follows, where we take the expectation wrt to the conditional distribution $P(\mathbf{Z}|\mathbf{D}, \theta')$, where θ' is the current estimate of parameters (from a prior or previous iteration) :

$$E_{P(\mathbf{Z}|\mathbf{D},\theta')}[PLL(\theta; \mathbf{z}, \mathbf{D})] = \sum_{\mathbf{z}\in\mathcal{Z}} P(\mathbf{Z}|\mathbf{D}, \theta') \log P(\mathbf{Z}, \mathbf{D}|\theta) \tag{C.1}$$

Here $\mathcal{Z}$ is the set of possible assignments to $\mathbf{Z}$. We rewrite the above as

$$Q(\theta, \theta') = \sum_{\mathbf{z}\in\mathcal{Z}} P(\mathbf{Z}|\mathbf{D}, \theta') \log P(\mathbf{D}|\mathbf{Z}, \theta) P(\mathbf{Z}|\theta) \tag{C.2}$$

We can assume that $P(\mathbf{Z}|\mathbf{D}, \theta') = P(Z_1, \cdots, Z_{|\mathbf{D}|}|\mathbf{D}, \theta') = \prod_{d=1}^{|D|} P(Z_d|\mathbf{x}_d, \theta')$. Similarly, $P(\mathbf{D}|\mathbf{Z}, \theta) = \prod_{d=1}^{|\mathbf{D}|} P(\mathbf{x}_d|Z_d, \theta)$, and $P(\mathbf{Z}|\theta) = \prod_{d=1}^{|\mathbf{D}|} P(Z_d|\theta)$, producing:

$$Q(\theta, \theta') = \sum_{\mathbf{z}\in\mathcal{Z}} \prod_{d_1=1}^{|\mathbf{D}|} P(Z_{d_1}|\mathbf{x}_{d_1}, \theta') \sum_{d=1}^{|\mathbf{D}|} \log P(\mathbf{x}_d|Z_d, \theta) P(Z_d|\theta) \tag{C.3}$$

$$Q(\theta,\theta') = \sum_{Z_1}\sum_{Z_2}\dots\sum_{Z_{|\mathbf{D}|}}\prod_{d_1=1}^{|\mathbf{D}|} P(Z_{d_1}|\mathbf{x}_{d_1},\theta')\sum_{d=1}^{|\mathbf{D}|}\log P(\mathbf{x}_d|Z_d,\theta)P(Z_d|\theta) \tag{C.4}$$

Taking the product inside the sum of d,

$$Q(\theta,\theta') = \sum_{Z_1}\sum_{Z_2}\dots\sum_{Z_{|\mathbf{D}|}}\sum_{d=1}^{|\mathbf{D}|}\left(\prod_{d_1=1}^{|\mathbf{D}|} P(Z_{d_1}|\mathbf{x}_{d_1},\theta')\right)\log P(\mathbf{x}_d|Z_d,\theta)P(Z_d|\theta) \tag{C.5}$$

In fact we can move the sum of d to be the outermost sum

$$Q(\theta,\theta') = \sum_{d=1}^{|\mathbf{D}|}\sum_{Z_1}\sum_{Z_2}\dots\sum_{Z_{|\mathbf{D}|}}\left(\prod_{d_1=1}^{|\mathbf{D}|} P(Z_{d_1}|\mathbf{x}_{d_1},\theta')\right)\log P(\mathbf{x}_d|Z_d,\theta)P(Z_d|\theta) \tag{C.6}$$

Let us consider one particular datapoint e and reorder the sums of Z_d :

$$\sum_{Z_e}\sum_{Z_1}\dots\sum_{Z_{e-1}}\sum_{Z_{e+1}}\dots\sum_{Z_{|\mathbf{D}|}}\left(\prod_{d_1=1}^{|\mathbf{D}|} P(Z_{d_1}|\mathbf{x}_{d_1},\theta')\right)\log P(\mathbf{x}_e|Z_e,\theta)P(Z_e|\theta)$$

Rearranging terms

$$\sum_{Z_e}\log P(\mathbf{x}_e|Z_e,\theta)P(Z_e|\theta)\left(\sum_{Z_{e-1}}\sum_{Z_{e+1}}\dots\sum_{Z_{|\mathbf{D}|}}\left(\prod_{d_1=2}^{|\mathbf{D}|} P(Z_{d_1}|\mathbf{x}_{d_1})\right)\left(\sum_{Z_1} P(Z_1|\mathbf{x}_1,\theta')\right)\right)$$

$\sum_{Z_1} P(Z_1|\mathbf{x}_1,\theta') = 1$. Similarly all the terms inside the outermost parenthesis become 1 leaving behind terms only for e^{th} datapoint, $\sum_{Z_e}\log P(\mathbf{x}_e|Z_e,\theta)P(Z_e|\theta)$. Thus we have

$$Q(\theta,\theta') = \sum_{d=1}^{|\mathbf{D}|}\sum_{Z_d} P(Z_d = k|\mathbf{x}_d,\theta')\log P(Z_d = k|\theta)P(\mathbf{x}_d|Z_d = k,\theta) \tag{C.7}$$

The $P(Z_d = k|\mathbf{x}_d,\theta')$ are the expected probabilities γ_{kd} of $\mathbf{x}_d$ being generated from the k^{th} model given the previous parameters θ'. Further, $P(Z_d = k|\theta) = \alpha_k$. Finally, $P(\mathbf{x}_d|Z_d = k,\theta) = P(\mathbf{x}_d|\theta_k)$

$$Q(\theta,\theta') = \sum_{d=1}^{|\mathbf{D}|}\sum_{Z_d}\gamma_{dk}\mathrm{log}\alpha_k P(\mathbf{x}_d|\theta_k) \tag{C.8}$$

Using our pseudo likelihood definition of $P(\mathbf{x}_d)$:

$$Q(\theta,\theta') = \sum_{d=1}^{|\mathbf{D}|}\sum_{Z_d}\gamma_{dk}\mathrm{log}\alpha_k \prod_{i=1}^{N} P(X_i = x_{id}|\mathbf{N}_{ki} = \mathbf{n}_{kdi},\theta_{ki}) \tag{C.9}$$

which decomposes into pseudo likelihood scores of a variable and its neighborhood:

$$Q(\theta,\theta') = \sum_{d=1}^{|\mathbf{D}|}\sum_{k=1}^{K}\gamma_{dk}(\mathrm{log}\alpha_k + \sum_{i=1}^{N}\mathrm{log}P(X_i = x_{id}|\mathbf{N}_{ki} = \mathbf{n}_{kdi},\theta_{ki})) \tag{C.10}$$

$$Q(\theta,\theta') = \sum_{k=1}^{K}\sum_{d=1}^{|\mathbf{D}|}\gamma_{dk}(\mathrm{log}\alpha_k + \sum_{i=1}^{N}\mathrm{log}P(X_i = x_{id}|\mathbf{N}_{ki} = \mathbf{n}_{kdi},\theta_{ki})) \tag{C.11}$$

$$Q(\theta,\theta') = \sum_{k=1}^{K}\sum_{d=1}^{|\mathbf{D}|}\gamma_{dk}(\sum_{i=1}^{N}\frac{\mathrm{log}\alpha_k}{N} + \sum_{i=1}^{N}\mathrm{log}P(X_i = x_{id}|\mathbf{N}_{ki} = \mathbf{n}_{kdi},\theta_{ki})) \tag{C.12}$$

$$Q(\theta,\theta') = \sum_{k=1}^{K}\sum_{d=1}^{|\mathbf{D}|}\gamma_{dk}(\sum_{i=1}^{N}\mathrm{log}\alpha_k^{\frac{1}{N}} P(X_i = x_{id}|\mathbf{N}_{ki} = \mathbf{n}_{kdi},\theta_{ki})) \tag{C.13}$$

Reordering the summations

$$Q(\theta,\theta') = \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{d=1}^{|\mathbf{D}|}\gamma_{dk}\mathrm{log}\alpha_k^{\frac{1}{N}} P(X_i = x_{id}|\mathbf{N}_{ki} = \mathbf{n}_{kdi},\theta_{ki}) \tag{C.14}$$

The sum decomposes over each variable, indicating we can optimize this sum by independently optimizing over each variable. Compare this with the score in Eq 6.15 we had prior to introducing Z_d, where it was not clear how to optimize over each variable:

$$\mathrm{log}P(\mathbf{D}|\mathcal{G},\theta_1,\cdots,\theta_K) = \sum_{d=1}^{|\mathbf{D}|}\log\left(\sum_{k=1}^{K}\alpha_k\left(\prod_{i=1}^{N}P(X_i = x_{di}|\mathbf{N}_{ki} = \mathbf{n}_{kdi},\theta_{ki})\right)\right) \tag{C.15}$$

Appendix D

Parameter tying in the conditional Gaussian mixture

In this approach, we force the dimensions of the weight vector $\mathbf{A}_{ki}$ to be the same for shared parents, where $\mathbf{A}_{ki}$ represents the weight vector for variable X_i in condition k. To derive ML estimates of $\mathbf{A}_{ki}$ we split the weight vector into the shared and unshared parts, and take derivatives with respective to these parts. Because the derivation holds for any X_i, we drop the subscript i for notational clarity. Thus, $\mathbf{A}_{\mathbf{ki}} = \mathbf{A}_k$. We write $\mathbf{A}_k = [\mathbf{B}_k\mathbf{C}]$, where $\mathbf{B}_k$ represents the weights for neighbors of X specific to condition k, and $\mathbf{C}$ represent the weight vector for the shared neighbors across conditions. Similarly we write $\mathbf{m}_{kj}$, which is the joint assignment to neighbors of X in condition k from datapoint j as, $\mathbf{m}_{kj} = [\mathbf{p}_{kj}\mathbf{q}_j]$, where $\mathbf{p}_{kj}$ represents the assignment to the condition k-specific neighbors of X and $\mathbf{q}_j$ represents the assignment to the shared neighbors of X. Putting this in the case of $k = \{1, 2\}$

- **ML estimate of shared parameters C**

$$\frac{\delta}{\delta \mathbf{C}} = \sum_{j=1}^{|\mathcal{D}|} \gamma_{j1} \left(\frac{\delta}{\delta \mathbf{B}} \frac{(x_{1j} - \mathbf{A}_1 \mathbf{m}_{1j} - b_1)^2}{-2\sigma_1} \right)$$

$$+\gamma_{j2}\left(\frac{\delta}{\delta \mathbf{C}}\frac{(x_{1j}-\mathbf{A}_2\mathbf{m}_{2j}-b_2)^2}{-2\sigma_2}\right)$$

We replace $\mathbf{A}_1\mathbf{m}_{1j}=\mathbf{B}_1\mathbf{p}_{1j}+\mathbf{C}\mathbf{q}_j$ and $\mathbf{A}_2\mathbf{m}_{2j}=\mathbf{B}_2\mathbf{p}_{1j}+\mathbf{C}\mathbf{q}_j$:

$$\frac{\delta}{\delta \mathbf{C}}=\sum_{j=1}^{|\mathcal{D}|}\gamma_{j1}\left(\frac{\delta}{\delta \mathbf{C}}\frac{(x_{1j}-\mathbf{B}_1\mathbf{p}_{1j}-\mathbf{C}\mathbf{q}_j-b_1)^2}{-2\sigma_1}\right)$$

$$+\gamma_{j2}\left(\frac{\delta}{\delta \mathbf{C}}\frac{(x_{1j}-\mathbf{B}_2\mathbf{p}_{2j}-\mathbf{C}\mathbf{q}_j-b_2)^2}{-2\sigma_2}\right)$$

$$=\sum_{j=1}^{|\mathcal{D}|}\gamma_{j1}\left(\frac{2(x_{1j}-\mathbf{B}_1\mathbf{p}_{1j}-\mathbf{C}\mathbf{q}_j-b_1)(-\mathbf{q}_j^\intercal)}{-2\sigma_1}\right)$$

$$+\gamma_{j2}\left(\frac{2(x_{1j}-\mathbf{B}_2\mathbf{p}_{2j}-\mathbf{C}\mathbf{q}_j-b_2)(-\mathbf{q}_j^\intercal)}{-2\sigma_2}\right)$$

Canceling the 2 and the signs and grouping terms with and without $\mathbf{C}$

$$=\sum_{j=1}^{|\mathcal{D}|}\gamma_{j1}\left(\frac{(x_{1j}-\mathbf{B}_1\mathbf{p}_{1j}-b_1)\mathbf{q}_j^\intercal}{\sigma_1}\right)+\gamma_{j2}\left(\frac{(x_{1j}-\mathbf{B}_2\mathbf{p}_{2j}-b_2)\mathbf{q}_j^\intercal}{\sigma_2}\right)$$

$$-\gamma_{j1}\left(\frac{\mathbf{C}\mathbf{q}_j\mathbf{q}_j^\intercal}{\sigma_1}\right)-\gamma_{j2}\left(\frac{\mathbf{C}\mathbf{q}_j\mathbf{q}_j^\intercal}{\sigma_2}\right)$$

Setting to 0 and solving for $\mathbf{C}$ gives

$$\mathbf{C}=\frac{\sum_{j=1}^{|\mathcal{D}|}\gamma_{j1}\left(\frac{(x_{1j}-\mathbf{B}_1\mathbf{p}_{1j}-b_1)\mathbf{q}_j^\intercal}{\sigma_1}\right)+\gamma_{j2}\left(\frac{(x_{1j}-\mathbf{B}_2\mathbf{p}_{2j}-b_2)\mathbf{q}_j^\intercal}{\sigma_2}\right)}{\sum_{j=1}^{|\mathcal{D}|}\frac{\gamma_{j1}\mathbf{q}_j\mathbf{q}_j^\intercal}{\sigma_1}+\frac{\gamma_{j2}\mathbf{q}_j\mathbf{q}_j^\intercal}{\sigma_2}}$$

- **ML estimate of unshared parameters** B_1

$$\frac{\delta}{\delta \mathbf{B}_1}=\sum_{j=1}^{|\mathcal{D}|}\gamma_{j1}\left(\frac{\delta}{\delta \mathbf{B}_1}\frac{(x_{1j}-\mathbf{B}_1\mathbf{p}_{1j}-\mathbf{C}\mathbf{q}_j-b_1)^2}{-2\sigma_1}\right)$$

$$+\gamma_{j2}\left(\frac{\delta}{\delta \mathbf{B}_1}\frac{(x_{1j}-\mathbf{B}_2\mathbf{p}_{2j}-\mathbf{C}\mathbf{q}_j-b_2)^2}{-2\sigma_2}\right)$$

$$= \sum_{j=1}^{|\mathcal{D}|} \gamma_{j1} \left(\frac{2(x_{1j} - \mathbf{B}_1 \mathbf{p}_{1j} - \mathbf{C}\mathbf{q}_j - b_1)(-\mathbf{p}_{1j}^{\mathsf{T}})}{-2\sigma_1} \right)$$

Grouping terms with and without $\mathbf{B}_1$

$$= \sum_{j=1}^{|\mathcal{D}|} \gamma_{j1} \left(\frac{(x_{1j} - \mathbf{C}\mathbf{q}_j - b_1)(\mathbf{p}_{1j}^{\mathsf{T}})}{\sigma_1} \right) - \sum_{j=1}^{|\mathcal{D}|} \frac{\gamma_{j1} \mathbf{B}_1 \mathbf{p}_{1j} \mathbf{p}_{1j}^{\mathsf{T}}}{\sigma_1}$$

Setting to 0 and solving for $\mathbf{B}_1$ gives the same estimate as in the completely unshared case

$$\mathbf{B}_1 = \frac{\sum_{j=1}^{|\mathcal{D}|} \gamma_{j1}(x_{1j} - \mathbf{C}\mathbf{q}_j - b_1)(\mathbf{p}_{1j}^{\mathsf{T}})}{\sum_{j=1}^{|\mathcal{D}|} \gamma_{j1} \mathbf{p}_{1j} \mathbf{p}_{1j}^{\mathsf{T}}}$$

Appendix E

Supplementary GO information of quiescent and non-quiescent populations

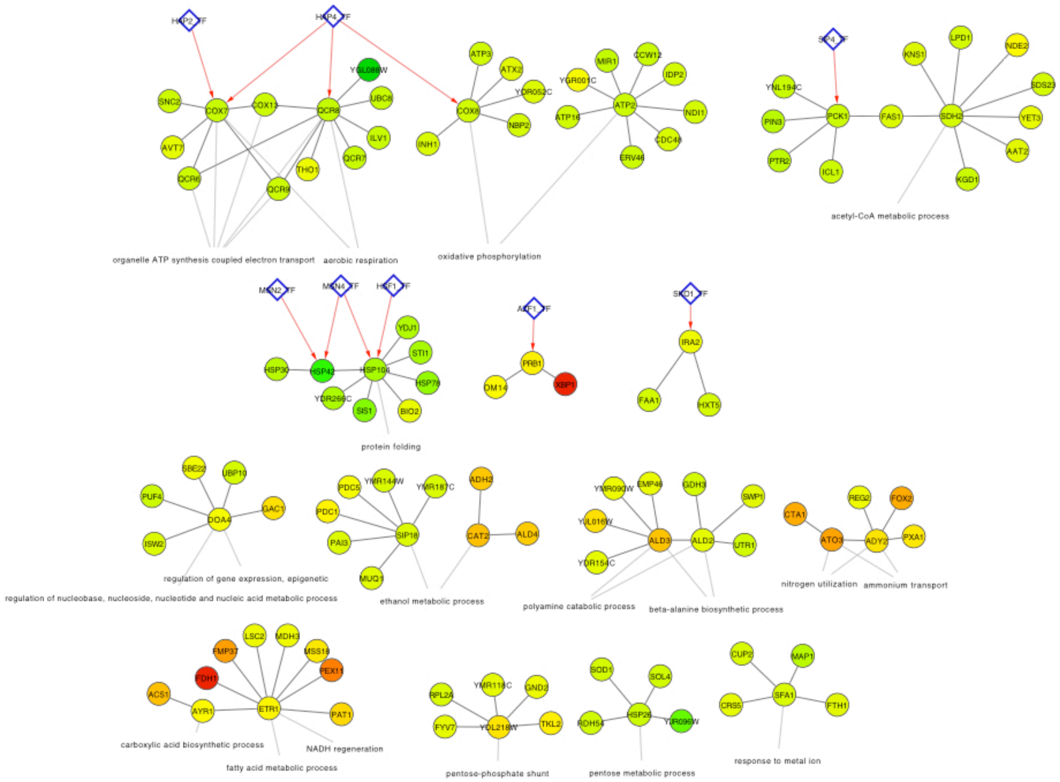


Figure E.1: GO processes and TF targets for subgraphs from the NIPD-inferred networks using the quiescent population. The text below each subgraph indicates the process. The diamonds represent the TFs. A TF is connected to the subgraph which is enriched in the targets of the TF. The circular nodes represent the genes in the network and color represents the extent of differential expression, red: up-regulated, green: down-regulated.

	INDEP	NIPD	MUG	PVEM	GC-COND	GC-CONDSH
			GOSLIM Process			
amino acid and derivative metabolic process	QNQ	QNQ		Q	Q	
carbohydrate metabolic process			Q	Q		
cell wall organization and biogenesis	Q					Q
cellular respiration	QNQ	QNQ	Q	QNQ	QNQ	QNQ
cofactor metabolic process	Q		Q		NQ	
generation of precursor metabolites and energy	QNQ	QNQ	QNQ	QNQ	QNQ	QNQ
lipid metabolic process	QNQ	QNQ		NQ	Q	Q
protein catabolic process					Q	
protein folding	QNQ	QNQ	QNQ			QNQ
response to chemical stimulus	Q		Q		Q	NQ
sporulation			Q	Q	Q	NQ
vesicle-mediated transport			Q	Q		
vitamin metabolic process		QNQ				NQ
			GOSLIM Function			
DNA binding	Q			NQ		QNQ
enzyme regulator activity	Q	Q	NQ		NQ	
ligase activity	Q	Q				
lyase activity	QNQ	QNQ	Q	NQ	Q	
oxidoreductase activity	QNQ	QNQ	Q	QNQ	QNQ	Q
peptidase activity	Q	Q				
protein binding	QNQ	QNQ	Q	Q		Q
transporter activity	QNQ	QNQ	QNQ	QNQ	QNQ	Q
			GOSLIM Component			
cell cortex		QNQ	Q			
cell wall		QNQ				
endomembrane system			Q			
endoplasmic reticulum	NQ			NQ		
mitochondrial envelope	QNQ	QNQ	Q	QNQ	QNQ	QNQ
peroxisome	Q	QNQ	Q	QNQ	QNQ	Q
plasma membrane	NQ		NQ	QNQ		
ribosome			NQ	NQ	NQ	Q
vacuole					Q	

Table E.1: GO Slim process, function and cellular component using the different condition-specific network learning algorithms. Q: Quiescent only, NQ: Non-quiescent only and QNQ: Quiescent and Non-quiescent.

aerobic respiration	GC-COND	YDR327W,WBP1,QCR6,COX13,QCR10,QCR8,CYC1
	MUG	UBC8,QCR6,YGL088W,QCR9,QCR8
	INDEP	YET3,NDE2,KGD1,SDH2,AAT2 QCR7,UBC8,THO1,YGL088W,QCR9,QCR8
	NIPD	QCR7,UBC8,THO1,ILV1,QCR6,YGL088W,COX13,QCR9,QCR8 QCR6,COX13,QCR9,AVT7,COX7,SNC2
	PVEM	HMRA2,TSC13,HNT1,RIP1,KGD1,SDH2 QCR7,UBC8,QCR9,QCR8,CYC1,GCD1
ammonium transport	GC-COND	OM14,ADY2,ATO3,ALD4 SHM1,ADY2,YRB1,BMH2,ATO3,YMR114C
	GC-CONDSH	ACS1,ADY2,ATO3,ADH2 ADY2,VCX1,BMH2,ATO3,YGR146C,RNR4
	INDEP	ADY2,ATO3,ADH2
	NIPD	ADY2,CTA1,ATO3 REG2,ADY2,ATO3,FOX2,PXA1
	PVEM	ACS1,OM14,ADY2,ATO3 ADY2,ATO3,ECI1,TEF1
organelle ATP synthesis coupled electron transport	GC-COND	COX13,QCR8,COX7 YDR327W,WBP1,QCR6,COX13,QCR10,QCR8,CYC1 RIP1,SDH2,COX8
	GC-CONDSH	AGX1,COX8,COX5A QCR6,COX13,QCR9,HXT4,QCR8,CYC1,ACO1,GTO3 ATP16,HNT1,SDH4,RIP1,FAS1,YKL199C,GAP1,SDH2,IDI1
	MUG	UBC8,QCR6,YGL088W,QCR9,QCR8 COX9,COX13,SIP4,MEF2,COX7
	NIPD	QCR7,UBC8,THO1,ILV1,QCR6,YGL088W,COX13,QCR9,QCR8 QCR6,QCR8,COX7 QCR9,QCR8,COX7 COX13,QCR8,COX7 QCR6,COX13,QCR9,AVT7,COX7,SNC2
	PVEM	HSP30,COX9,COX8 ATP3,COX9,INH1,TIM13,COX8,ATP20 QCR7,UBC8,QCR9,QCR8,CYC1,GCD1
oxidative phosphoryl-ation	GC-COND	ATP1,CRF1,AFG1,ATP2,PUS5,NDI1,TAF9,GAS5
	INDEP	ATP3,INH1,COX8 QCR6,COX13,AVT7,COX7 QCR7,UBC8,THO1,YGL088W,QCR9,QCR8
	NIPD	ERV46,ATP16,CDC48,YGR001C,MIR1,ATP2,CCW12,IDP2,NDI1 ATP3,INH1,NBP2,COX8,YOR052C,ATX2
	PVEM	ATP3,COX9,INH1,TIM13,COX8,ATP20 ECM32,QCR6,COX13,COX7,YOR292C,ATP15
nitrogen utilization	GC-COND	OM14,ADY2,ATO3,ALD4
	GC-CONDSH	ACS1,ADY2,ATO3,ADH2
	INDEP	ADY2,ATO3,ADH2
	NIPD	ADY2,CTA1,ATO3 REG2,ADY2,ATO3,FOX2,PXA1
	PVEM	ACS1,OM14,ADY2,ATO3 ADY2,ATO3,ECI1,TEF1
polyamine catabolic process	NIPD	GDH3,UTR1,SWP1,ALD3,ALD2 YDR154C,YJL016W,EMP46,YMR090W,ALD3,ALD2

carboxylic acid biosynthetic process	MUG	ETR1,FMP37,AYR1
	INDEP	ETR1,YIL039W,AYR1,PEX11,MSS18
	NIPD	ACS1,ETR1,AYR1
	PVEM	ETR1,FMP37,SOD2,AYR1,YHM2
	GC-CONDSH	YBR016W,TAT1,BMH1,YJL217W,YHM2,PUT4,ODC1
monocarboxylic acid metabolic process	GC-CONDSH	YAT1,YAT2,JEN1
		POT1,YIR016W,PHO84,HRP1,PEX11
		YAT1,YBR014C,YAT2,NUP82,OPI3,ZRT2,YLR179C,ERG10
ethanol metabolic process	NIPD	CAT2,ADH2,ALD4
		MUQ1,PDC1,PDC5,YMR144W,PAI3,SIP18,YMR187C
acetyl-CoA metabolic process	NIPD	YET3,NDE2,LPD1,SDS23,KGD1,FAS1,KNS1,SDH2,AAT2
carboxylic acid transport	GC-CONDSH	YBR016W,TAT1,BMH1,YJL217W,YHM2,PUT4,ODC1
pentose metabolic process	NIPD	HSP26,RDH54,SOL4,YJR096W,SOD1
pentose-phosphate shunt	GC-CONDSH	TKL2,KNH1,YDL218W,RPS13,CHO2,GND2
	NIPD	TKL2,YDL218W,RPL2A,GND2,FYV7,YMR118C
pentose-phosphate shunt, oxidative branch	GC-COND	YAL061W,VHS1,GRH1,
		SOL4,GND2,YKL151C,GAD1
protein folding	MUG	HSP78,PMP3,HSP104,SIS1,STI1
	NIPD	HSP42,HSP78,YDR266C,BIO2,HSP104,SIS1,YDJ1,STI1
	PVEM	OST4,HSP78,SSA2,HSP104,SIS1,MRPS18,STI1
protein deubiquitination	INDEP	DOA4,PUF4,UBP10

Table E.2: GO processes in which subgraphs identified by different methods are enriched. The first column shows the process, the second column the method, and the third column describes the genes within each of the subgraphs.

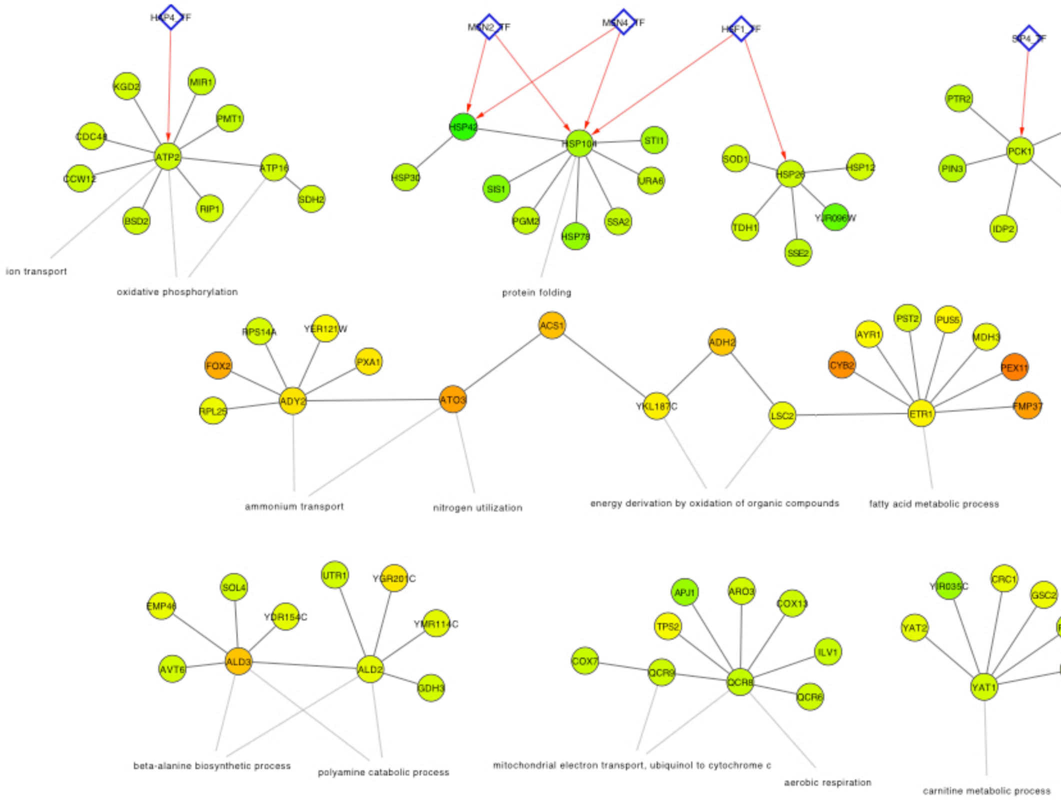


Figure E.2: GO processes and TF targets for subgraphs from the NIPD-inferred networks using the non-quiescent population. Legend is similar to Fig E.1

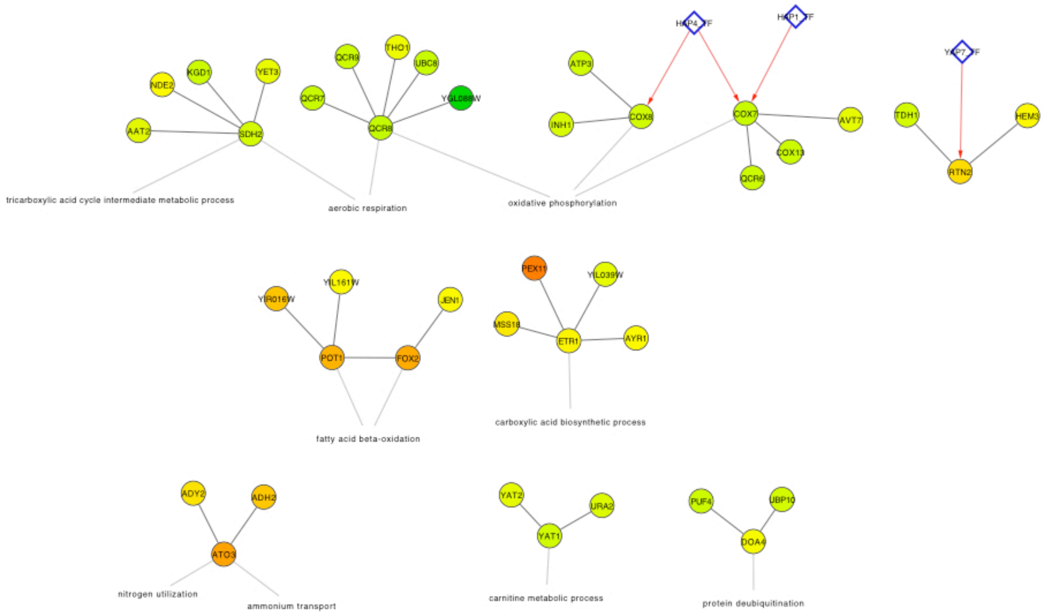


Figure E.3: GO processes and TF targets for subgraphs from the INDEP-inferred networks using the quiescent population. The text below each subgraph indicates the process. The diamonds represent the TFs. A TF is connected to the subgraph which is enriched in the targets of the TF. The circular nodes represent the genes in the network and color represents the extent of differential expression, red: up-regulated, green: down-regulated.

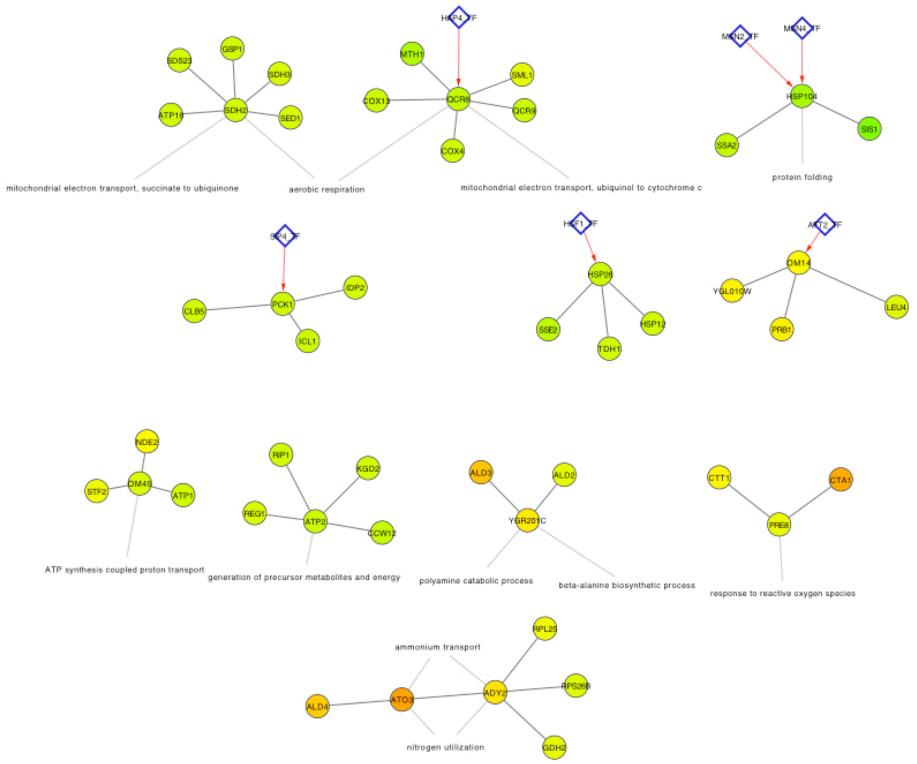


Figure E.4: GO processes and TF targets for subgraphs from the INDEP-inferred networks using the non-quiescent population. Legend is similar to Fig E.3

Glossary

condition
A global, discrete variable that influences the changes in gene, protein and other macromolecular expression at a genome-wide scale.

microarrays
A glass slide on which the complete genome of an organism is printed.

gene expression
mRNA level of a particular gene used to quantify the activity level of a gene.

genome
The collection of the functional elements of a DNA, typically referring to the complete set of genes.

proteome
The collection of all proteins in a cell.

metabolome
The collection of all metabolites within a cell.

phenotype
Literally derived from the words “phenomenon” and describes physical, observable characteristic of an organism in response to a genetic perturbation. Ideally, we want identify the gene *causing* a phenotype.

transcription factors
Proteins which activate or repress, often combinatorially, the process of transcription – making of mRNA from the

DNA sequence of gene.

master regulator

Transcription factors that are generic and control a variety of target genes, often involved in developmental regulatory networks or in stress response.

physical network

A network of bio-chemical entities (genes, proteins, metabolites), with edges representing interactions where the nodes physically interact. Examples are two proteins interacting to form a protein dimer.

Gene ontology

Controlled vocabulary of hierarchically arranged terms a used to annotate genes in different model organisms.

ortholog

A gene from one species is said to be an ortholog of a gene from another species, if they have high sequence similarity and have been inherited from a common ancestor.

functional network

A network of genes, and other macro-molecules, where the edges correspond to a functional or phenomenological relationship. The relationship is often specified by the amount of statistical dependence between the network nodes.

network inference

The problem of identifying the structure (the edges) and function (type of statistical dependency) among network nodes using measured attributes of the network nodes.

graph

A mathematical object composed of a set of nodes (genes, or random variables) and a set of edges connecting the nodes. Undirected graphs are those where edges don't have directionality and directed graphs are those where edges do have directionality. Nodes of the graph are also called vertices.

Markov blanket

A concept used in probabilistic graphical models, used to specify the immediate neighborhood of a node.

Shortest path

A path between two vertices of an network is a set of edges which connect one vertex to another.

Connected component

A concept from graph theory, which specifies the part of the graph where every vertex is connected, or has a path, to every other vertex.

Vertex degree

The number of immediate neighbors of a vertex.

Indegree

A concept in a directed graph defined for each vertex, v. The indegree is the number of nodes with edges pointing to v.

Outdegree

A concept in a directed graph defined for each vertex, v. The outdegree is the number of nodes with edges pointing away from v.

Hidden variables

Random variables that are part of the model, but do not correspond to anything that is measurable. Typically, hidden variables influence the state of the observed variables: the variables corresponding to measurable quantities.

Expectation maximization

An iterative meta-algorithm used often in machine learning to learn parameters of a model with hidden variables. Hidden variables are assigned their expected values and

	parameters are estimated using the combined set of expected and observed values of the variables.
Probabilistic graphical models	A class of models from machine learning which are used to represent statistical relationships among multiple entities. The model has two components: a graph structure describing which entities are dependent on each other, and a set of probabilistic functions describing the type of statistical dependency. Some examples of types are linear, non-linear, pairwise, higher-order etc.
Markov random fields	A class of probabilistic graphical models in which the graph structure is undirected. These models can capture cyclic dependencies.
Bayesian networks	A type of probabilistic graphical model in which the graph structure is directed. The graph structure is constrained to prevent cycles, and therefore cannot capture cyclic dependencies.
EM	Expectation maximization
PLL	Pseudo log likelihood.
MRF	Markov random field.
MBS	Markov Blanket Search.
CIN	Cluster and Infer Networks.
PGM	Probabilistic graphical model.
NIPD	Network inference with pooling data.

INDEP	INDEPendent learner of graphs from two or more conditions, one per condition.
MUG	Mixture of graphs.
PVEM	Per-variable Expectation Maximization model.
GC-COND	Graph component with CONDditional distributions.
GC-CONDSH	GC-COND model with parameter tying-based data sharing.

References

[1] Pieter Abbeel, Daphne Koller, and Andrew Y. Ng. Learning factor graphs in polynomial time and sample complexity. *JMLR*, 7:1743–1788, 2006.

[2] Reka Albert and Albert-Laszlo Barabasi. Topology of evolving networks: local events and universality. *Phys. Rev. Lett.*, 85:5234–5237, 2000.

[3] C. Allen, S. Büttner, A. D. Aragon, J. A. Thomas, O. Meirelles, J. E. Jaetao, D. Benn, S. W. Ruby, M. Veenhuis, F. Madeo, and M. Werner-Washburne. Isolation of quiescent and nonquiescent cells from yeast stationary-phase cultures. *J Cell Biol*, 174(1):89–100, July 2006.

[4] A. D. Aragon, G. A. Quiñones, E. V. Thomas, S. Roy, and M. Werner-Washburne. Release of extraction-resistant mrna in stationary phase saccharomyces cerevisiae produces a massive increase in transcript abundance in response to stress. *Genome Biol*, 7(2), 2006.

[5] Anthony D. Aragon, Angelina L. Rodriguez, Osorio Meirelles, Sushmita Roy, George S. Davidson, Chris Allen, Ray Joe, Phillip Tapia, Don Benn, and Margaret Werner-Washburne. Characterization of differentiated quiescent and non-quiescent cells in yeast stationary-phase cultures. *Molecular Biology of the Cell*, 2008.

[6] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet*, 25(1):25–29, May 2000.

[7] Janne Aukia, Samuel Kaski, and Janne Sinkkonen. Inferring vertex properties from topology in large network. In *NIPS 2007 Workshop on Statistical Network Models*, 2007.

[8] Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego di Bernardo. How to infer gene networks from expression profile. *Molecular Systems Biology*, 3(78), 2007.

[9] Z. Bar-Joseph, G. K. Gerber, T. I. Lee, N. J. Rinaldi, J. Y. Yoo, F. Robert, D. B. Gordon, E. Fraenkel, T. S. Jaakkola, R. A. Young, and D. K. Gifford. Computational discovery of gene modules and regulatory networks. *Nat Biotechnol*, 21(11):1337–1342, November 2003.

[10] A. Battle, E. Segal, and D. Koller. Probabilistic discovery of overlapping cellular processes and their regulation. *J Comput Biol*, 12(7):909–927, September 2005.

[11] J. Beirlant, E. J. Dudewicz, L. Györfi, and E. C. Meulen. Nonparametric entropy estimation: An overview. *International Journal of the Mathematical Statistics Sciences*, 6:17–39, 1997.

[12] Archana Belle, Amos Tanay, Ledion Bitincka, Ron Shamir, and Erin K. O'Shea. Quantification of protein half-lives in the budding yeast proteome. *PNAS*, 103(35):13004–13009, August 2006.

[13] S. Bergmann, J. Ihmels, and N. Barkai. Similarities and differences in genome-wide expression data of six organisms. *PLoS Biol*, 2(1), January 2004.

[14] Sven Bergmann, Jan Ihmels, and Naama Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 67(3 Pt 1), March 2003.

[15] Julian Besag. Efficiency of pseudolikelihood estimation for simple gaussian fields. *Biometrika*, 64(3):616–618, December 1977.

[16] Alexandre Blais and Brian David Dynlacht. Constructing transcriptional regulatory networks. *Genes and Development*, 19, 2005.

[17] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[18] Richard Bonneau, David J Reiss, Paul Shannon, Marc Facciotti, Leroy Hood, Nitin S Baliga, and Vesteinn Thorsson. The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biology*, 2006.

[19] R. R. Bouckaert. Probabilistic network construction using the minimum description length principle. *Lecture Notes in Computer Science*, 747:41–??, 1993.

[20] Elizabeth I. Boyle, Shuai Weng, Jeremy Gollub, Heng Jin, David Botstein, Michael M. Cherry, and Gavin Sherlock. Go::termfinder-open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics*, 20(18):3710+, 2005.

[21] Rainer Breitling, Anna Amtmann, and Pawel Herzyk. Graph-based iterative group analysis enhances microarray interpretation. *BMC Bioinformatics*, 5(1):100+, July 2004.

[22] Michael J. Buck and Jason D. Lieb. A chromatin-mediated mechanism for specification of conditional transcription factor targets. *Nat Genet*, 38(12):1446–1451, December 2006.

[23] Brian W W. Busser, Martha L L. Bulyk, and Alan M M. Michelson. Toward a systems-level understanding of developmental regulatory networks. *Current opinion in genetics & development*, October 2008.

[24] Robert Castelo and Alberto Roverato. A robust procedure for gaussian graphical model search from microarray data with larger than . *Journal of Machine Learning Research*, 6:2621–2650, 2006.

[25] Juan I. Castrillo, Leo A. Zeef, David C. Hoyle, Nianshu Zhang, Andrew Hayes, David C. J. Gardner, Michael J. Cornell, June Petty, Luke Hakes, Leanne Wardleworth, Bharat Rash, Marie Brown, Warwick B. Dunn, David Broadhurst, Kerry O'Donoghue, Svenja S. Hester, Tom P. J. Dunkley, Sarah R. Hart, Neil Swainston, Peter Li, Simon J. Gaskell, Norman W. Paton, Kathryn S. Lilley, Douglas B. Kell, and Stephen G. Oliver. Growth control of the eukaryote cell: a systems biology study in yeast. *Journal of Biology*, 6:4+, April 2007.

[26] Jeffrey T. Chang, Carlos Carvalho, Seiichi Mori, Andrea H. Bild, Michael L. Gatza, Quanli Wang, Joseph E. Lucas, Anil Potti, Phillip G. Febbo, Mike West, and Joseph R. Nevins. A genomic strategy to elucidate modules of oncogenic pathway signaling networks. *Molecular Cell*, 34(1):104–114, April 2009.

[27] Shann C. Chen, Geoffrey J. Gordon, and Robert F. Murphy. Graphical models for structured classification, with an application to interpreting images of protein subcellular location patterns. *J. Mach. Learn. Res.*, 9:651–682, 2008.

[28] David M. Chickering, Dan Geiger, and David Heckerman. Learning Bayesian Networks is NP-Hard. Technical Report MSR-TR-94-17, Microsoft research, November 1994.

[29] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

[30] Han-Yu Chuang, Eunjung Lee, Yu-Tsueng Liu, Doheon Lee, and Trey Ideker. Network-based classification of breast cancer metastasis. *Mol Syst Biol*, 3, October 2007.

[31] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 09(4):309–347, October 1992.

[32] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[33] Eric H. Davidson. *The regulatory genome : gene regulatory networks in development and evolution*. Academic, Burlington, MA, 2006.

[34] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J Comput Biol*, 9(1):67–103, 2002.

[35] Amol Deshpande, Minos N. Garofalakis, and Michael I. Jordan. Efficient stepwise selection in decomposable models. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 128–135, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[36] Karthik Devarajan. Nonnegative matrix factorization: An analytical and interpretive tool in computational biology. *PLoS Comput Biol*, 4(7):e1000029+, July 2008.

[37] Radu Dobrin, Jun Zhu, Cliona Molony, Carmen Argman, Mark Parrish, Sonia Carlson, Mark Allan, Daniel Pomp, and Eric Schadt. Multi-tissue coexpression networks reveal unexpected subnetworks associated with disease. *Genome Biology*, 10:R55+, May 2009.

[38] Daniel Eaton and Kevin Murphy. Belief net structure learning from uncertain interventions. *JMLR Special Topic on Causality*, 2008.

[39] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A*, 95(25):14863–14868, December 1998.

[40] Nir Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303:799–805, 2004.

[41] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309, 1999.

[42] Nir Friedman and Daphne Koller. Being bayesian about network structure. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 201–210, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[43] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. *Journal of Comp. Biol.*, 7(3-4):601–620, 2000.

[44] Nir Friedman, Iftach Nachman, and Dana Pe'er. Learning bayesian network structure from massive datasets: The sparse candidate algorithm. In *Uncertainty in Artificial Intelligence*, 1999.

[45] Thomas Gaertner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop*, 2003.

[46] Timothy S. Gardner, Diego di Bernardo, David Lorenz, and James J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 2003.

[47] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell*, 11(12):4241–4257, December 2000.

[48] Audrey P. Gasch and Margaret Werner-Washburne. The genomics of yeast responses to environmental stress and starvation. *Functional & integrative genomics*, 2(4-5):181–192, September 2002.

[49] I. Gat-Viks and R. Shamir. Refinement and expansion of signaling pathways: the osmotic response network in yeast. *Genome Res*, 17(3):358–367, March 2007.

[50] I. Gat-Viks, A. Tanay, and R. Shamir. Factor graph network models for biological systems. In *RECOMB*, 2005.

[51] Irit Gat-Viks, Amos Tanay, Daniela Raijman, and Ron Shamir. A probabilistic methodology for integrating knowledge and experiments on biological networks. *Journal of Computational Biology*, 13(2):165–181, March 2006.

[52] Dan Geiger and David Heckerman. Advances in probabilistic reasoning. In *Proceedings of the seventh conference (1991) on Uncertainty in artificial intelligence*, pages 118–126, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[53] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 1995.

[54] Tara A. Gianoulis, Jeroen Raes, Prianka V. Patel, Robert Bjornson, Jan O. Korbel, Ivica Letunic, Takuji Yamada, Alberto Paccanaro, Lars J. Jensen, Michael Snyder, Peer Bork, and Mark B. Gerstein. Quantifying environmental adaptation of

metabolic pathways in metagenomics. *Proceedings of the National Academy of Sciences*, 106(5):1374–1379, February 2009.

[55] Joseph V. Gray, Gregory A. Petsko, Gerald C. Johnston, Dagmar Ringe, Richard A. Singer, and Margaret Werner-Washburne. ”sleeping beauty”: Quiescence in saccharomyces cerevisiae. *Microbiol. Mol. Biol. Rev.*, 68(2):187–206, June 2004.

[56] D. Greenbaum, C. Colangelo, K. Williams, and M. Gerstein. Comparing protein abundance and mrna expression levels on a genomic scale. *Genome Biol*, 4(9), 2003.

[57] Elizabeth A. Grice, Heidi H. Kong, Sean Conlan, Clayton B. Deming, Joie Davis, Alice C. Young, Nisc Comparative Sequencing Program, Gerard G. Bouffard, Robert W. Blakesley, Patrick R. Murray, Eric D. Green, Maria L. Turner, and Julia A. Segre. Topographical and temporal diversity of the human skin microbiome. *Science*, 324(5931):1190–1192, May 2009.

[58] Joshua A. Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *RECOMB*, 2007.

[59] U. Gueldener, M. Muensterkoetter, G. Kastenmueller, N. Strack, J. van Helden, C. Lemer, J. Richelles, S. J. Wodak, J. Garcia-Martinez, J. E. Perez-Ortin, H. Michael, A. Kaps, E. Talla, B. Dujon, B. Andre, J. L. Souciet, J. De Montigny, E. Bon, C. Gaillardin, and H. W. Mewes. Cygd: the comprehensive yeast genome database. *Nucleic Acids Res*, 33(Database issue), January 2005.

[60] A. Hamosh, A. F. Scott, J. S. Amberger, C. A. Bocchini, and V. A. McKusick. Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Res*, 33 Database Issue, January 2005.

[61] Christopher T. Harbison, D. Benjamin Gordon, Tong Ihn Lee, Nicola J. Rinaldi, Kenzie D. Macisaac, Timothy W. Danford, Nancy M. Hannett, Jean-Bosco Tagne, David B. Reynolds, Jane Yoo, Ezra G. Jennings, Julia Zeitlinger, Dmitry K. Pokholok, Manolis Kellis, P. Alex Rolfe, Ken T. Takusagawa, Eric S. Lander, David K. Gifford, Ernest Fraenkel, and Richard A. Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 2004.

[62] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, August 2001.

[63] David Heckerman. A Tutorial on Learning Bayesian Networks. Technical Report MSR-TR-95-06, Microsoft research, March 1995.

[64] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Myers Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[65] David Heckerman, Dan Geiger, and David M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994.

[66] Reimar Hofmann and Volker Tresp. Nonlinear markov networks for continuous variables. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 521–527, Cambridge, MA, USA, 1998. MIT Press.

[67] S Hoops, S Sahle, R Gauges, C Lee, J Pahle, N Simus, M Singhal, L Xu, P Mendes, and U Kummer. Copasi – a complex pathway simulator. *Bioinformatics*, 22:3067–3074, 2006.

[68] Funchun Huang and Yosihiko Ogata. Generalized pseudo-likelihood estimates for markov random fields on lattice. *Annals of the Institute of Statistical Mathematics*, 54(1):1–18, March 2004.

[69] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, July 2000.

[70] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffry E. Hinton. Adaptive mixture of local experts. *Neural Computation*, 3:79–87, 1991.

[71] S. Jbabdi, M. W. Woolrich, and T. E. Behrens. Multiple-subjects connectivity-based parcellation using hierarchical dirichlet process mixture models. *NeuroImage*, 44(2):373–384, January 2009.

[72] Rice J. Jeremy, Tu Yuhai, and Stolovitzky Gustavo. Reconstructing biological networks using conditional correlation analysis. *Bioinformatics*, 21(6):765–773, March 2005.

[73] Michael I. Jordan. *Learning in graphical models*. MIT Press, 1998.

[74] Naftali Kaminski and Ziv Bar-Joseph. A patient-gene model for temporal expression profiles in clinical studies. In *Research in Computational Molecular Biology*, pages 69–82. Springer Verlag, 2006.

[75] Nadav Kashtan and Uri Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773–13778, September 2005.

[76] Manolis Kellis, Nick Patterson, Matthew Endrizzi, Bruce Birren, and Eric S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423(6937):241–254, May 2003.

[77] Hyunsoo Kim, William Hu, and Yuval Kluger. Unraveling condition specific gene transcriptional regulatory networks in saccharomyces cerevisiae. *BMC Bioinformatics*, 2006.

[78] Hiroaki Kitano. Systems biology: A brief overview. *Science*, 295(5560):1662–1664, March 2002.

[79] Hiroyuki Kurata, Nana Matoba, and Natsumi Shimizu. CADLIVE for constructing a large-scale biochemical network based on a simulation-directed notation and its application to yeast cell cycle. *Nucl. Acids Res.*, 31(14):4071–4084, 2003.

[80] Steffen L. Lauritzen. *Graphical Models*. Oxford Statistical Science Series. Oxford University Press, New York, USA, July 1996.

[81] Neil Lawrence, Guido Sanguinetti, and Magnus Rattray. Modelling transcriptional regulation using gaussian processes. In *NIPS 2006*, 2006.

[82] Neil D. Lawrence and John C. Platt. Learning to learn with the informative vector machine. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, pages 65+, New York, NY, USA, 2004. ACM.

[83] I. Lee, S. V. Date, A. T. Adai, and E. M. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306(5701):1555–1558, November 2004.

[84] Su-In Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of markov networks using l_1-regularization. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 817–824. MIT Press, Cambridge, MA, 2007.

[85] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J. B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298(5594):799–804, October 2002.

[86] P. Lesage, X. Yang, and M. Carlson. Yeast snf1 protein kinase interacts with sip4, a c6 zinc cluster transcriptional activator: a new role for snf1 in the glucose response. *Molecular and cellular biology*, 16(5):1921–1928, May 1996.

[87] Fan Li and Yiming Yang. Using modified lasso regression to learn large undirected graphs in a probabilistic framework. In *AAAI*, pages 801–806, 2005.

[88] James Long and Mitchell Roth. Synthetic microarray data generation with range and nemo. *Bioinformatics*, November 2007.

[89] P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble. Investigating semantic similarity measures across the gene ontology: the relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–1283, July 2003.

[90] Kenzie Macisaac, Ting Wang, D. Benjamin Gordon, David Gifford, Gary Stormo, and Ernest Fraenkel. An improved map of conserved regulatory sites for saccharomyces cerevisiae. *BMC Bioinformatics*, 7(1):113+, March 2006.

[91] A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano. Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, (Suppl 1): S7, 2005.

[92] Martin, Shawn, Zhang, Zhaoduo, Martino, Anthony, Faulon, and Jean-Loup. Boolean dynamics of genetic regulatory networks inferred from microarray time series data. *Bioinformatics*, 23(7):866–874, April 2007.

[93] Shawn Martin, Diana Roe, and Jean-Loup Falon. Predicting protein-protein interations using signature products. *Bioinformatics*, 21(2):218–226, 2005.

[94] M. Juanita Martinez, Sushmita Roy, Amanda B. Archuletta, Peter D. Wentzell, Sonia S. Anna-Arriola, Angelina L. Rodriguez, Anthony D. Aragon, Gabriel A. Quinones, Chris Allen, and Margaret Werner-Washburne. Genomic analysis of stationary-phase and exit in saccharomyces cerevisiae: Gene expression and identification of novel essential genes. *Mol. Biol. Cell*, 15(12):5295–5305, December 2004.

[95] Sergei Maslov and Kim Sneppen. Computational architecture of the yeast regulatory network. *Physical Biology*, 2:s94–s100, 2005.

[96] Mark I. Mccarthy, Goncalo R. Abecasis, Lon R. Cardon, David B. Goldstein, Julian Little, John P. A. Ioannidis, and Joel N. Hirschhorn. Genome-wide association studies for complex traits: consensus, uncertainty and challenges. *Nat Rev Genet*, 9(5):356–369, May 2008.

[97] Megan N N. McClean, Areez Mody, James R R. Broach, and Sharad Ramanathan. Cross-talk and decision making in map kinase pathways. *Nat Genet*, January 2007.

[98] Rachel P. Mccord, Michael F. Berger, Anthony A. Philippakis, and Martha L. Bulyk. Inferring condition-specific transcription factor function from dna binding and gene expression data. *Mol Syst Biol*, 3, 2007.

[99] Marina Meila and Michael I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

[100] Pedro Mendes, Wei Sha, and Keying Ye. Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19:122–129, 2003.

[101] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, October 2002.

[102] Vlad I. Morariu, Balaji Vasan Srinivasan, Vikas C. Raykar, Ramani Duraiswami, and Larry S. Davis. Automatic online tuning for fast gaussian summation. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[103] Tatiana V. Morozova, Robert R. Anholt, and Trudy F. Mackay. Transcriptional response to alcohol exposure in drosophila melanogaster. *Genome biology*, 7(10):R95+, October 2006.

[104] Mukund Narasimhan and Jeff Bilmes. Pac-learning bounded tree-width graphical models. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 410–417, Arlington, Virginia, United States, 2004. AUAI Press.

[105] Jennifer Neville and David Jensen. Relational dependency networks. *J. Mach. Learn. Res.*, 8:653–692, 2007.

[106] Alexandru Niculescu-Mizil and Rich Caruana. Inductive transfer for bayesian network structure learning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[107] Rupert D. Paget. *Nonparametric Markov random field models for natural texture images*. PhD thesis, University of Queensland, 1999.

[108] Wei Pan. A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. *Bioinformatics*, 18(4):546–554, April 2002.

[109] Dana Pe'er, Amos Tanay, and Aviv Regev. Minreg: A scalable algorithm for learning parsimonious regulatory networks in yeast and mammals. *J. Mach. Learn. Res.*, 7:167–189, 2006.

[110] Yuan Qi and Hui Ge. Modularity and dynamics of cellular networks. *PLoS Computational Biology*, 2(12):e174+, December 2006.

[111] John Rachlin, Dikla D. Cohen, Charles Cantor, and Simon Kasif. Biological context networks: a mosaic view of the interactome. *Mol Syst Biol*, 2, November 2006.

[112] Matthew V. Rockman. Reverse engineering the genotype-phenotype map with natural genetic variation. *Nature*, 456(7223):738–744, December 2008.

[113] Rokhlenko, Oleg, Wexler, Ydo, Yakhini, and Zohar. Similarities and differences of gene expression in yeast stress conditions. *Bioinformatics*, 23(2):e184–e190, January 2007.

[114] Jan Roman and Thomas Gaertner. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*, 2003.

[115] Sushmita Roy, Margaret Werner-Washburne, and Terran Lane. A system for generating transcription regulatory networks with combinatorial control of transcription. *Bioinformatics (Oxford, England)*, April 2008.

[116] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 2005.

[117] Heladia Salgado, Socorro Gama-Castro, Martin Peralta-Gil, Edgar Diaz-Peredo, Fabiola Sanchez-Solano, Alberto Santos-Zavaleta, Irma Martinez-Flores, Veronica Jimenez-Jacinto, Cesar Bonavides-Martinez, Juan Segura-Salazar, Agustino Martinez-Antonio, and Julio Collado-Vides. Regulondb (version 5.0): Escherichia coli k-12 transcriptional regulatory network, operon organization, and growth conditions. *Nucleic Acids Research*, 34:D394, 2006.

[118] Thomas Sandmann, Charles Girardot, Marc Brehme, Waraporn Tongprasit, Viktor Stolc, and Eileen E. M. Furlong. A core transcriptional network for early mesoderm development in drosophila melanogaster. *Genes & Development*, 21(4):436–449, February 2007.

[119] Guido Sanguinetti, Josselin Noirel, and Phillip C. Wright. Mmg: a probabilistic tool to identify submodules of metabolic pathways. *Bioinformatics*, 24(8):1078–1084, April 2008.

[120] Maria J. Schilstra and Hamid Bolouri. The logic of gene regulation. In *Poster abstract for Third International Conference on Systems Biology*, 2002.

[121] Mark Schmidt, Alexandru Niculescu-Mizil, and Kevin Murphy. Learning graphical model structure using l1-regularization paths. In *Twenty second international conference on AI*, 2007.

[122] Anton Schwaighofer, Mathäus Dejori, Volker Tresp, and Martin Stetter. Structure learning with nonparametric decomposable models. In *ICANN (1)*, pages 119–128, 2007.

[123] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat Genet*, 34(2):166–176, June 2003.

[124] Eran Segal, Nir Friedman, Naftali Kaminski, Aviv Regev, and Daphne Koller. From signatures to models: understanding cancer using microarrays. *Nature genetics*, 37 Suppl, June 2005.

[125] Eran Segal, Dana Pe'er, Aviv Regev, Daphne Koller, and Nir Friedman. Learning module networks. *Journal of Machine Learning Research*, 6:557–588, April 2005.

[126] Victor Spirin and Leonid A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123–12128, 2003.

[127] P. Spirtes, C. Glymour, and R. Scheines. Constructing bayesian network models of gene expression networks from microarray data, 2000.

[128] M. Steffen, A. Petti, J. Aach, P. D'haeseleer, and G. Church. Automated modelling of signal transduction networks. *BMC Bioinformatics*, 3, November 2002.

[129] T. Stein, J. Kricke, D. Becher, and T. Lisowsky. Azf1p is a nuclear-localized zinc-finger protein that is preferentially expressed under non-fermentative growth conditions in saccharomyces cerevisiae. *Current genetics*, 34(4):287–296, October 1998.

[130] J. D. Storey and R. Tibshirani. Statistical significance for genomewide studies. *Proc Natl Acad Sci U S A*, 100(16):9440–9445, August 2003.

[131] Joshua M. Stuart, Eran Segal, Daphne Koller, and Stuart K. Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302(5643):249–255, October 2003.

[132] Joe Suzuki. Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the b & b technique. In *International Conference on Machine Learning*, pages 462–470, 1996.

[133] Amos Tanay, Roded Sharan, Martin Kupiec, and Ron Shamir. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2981–2986, March 2004.

[134] Bo Thiesson, Christopher Meek, David M. Chickering, and David Heckerman. Learning mixtures of dag models. In *Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference (1998)*, pages 504–513. Morgan Kaufmann Publishers, 1998.

[135] Lidia Tomás-Cobos, Laura Casadomé, Glòria Mas, Pascual Sanz, and Francesc Posas. Expression of the hxt1 low affinity glucose transporter requires the coordinated activities of the hog and glucose signalling pathways. *The Journal of biological chemistry*, 279(21):22010–22019, May 2004.

[136] D. P. Tuck, H. M. Kluger, and Y. Kluger. Characterizing disease states from topological properties of transcriptional regulatory networks. *BMC Bioinformatics*, 7, 2006.

[137] E. P. van Someren, L. F. A. Wessels, and M. J. T. Reinders. Linear modeling of genetic networks from experimental data. In *Proceedings of the Eighth International Conference on ISMB*, pages 355–366, 2000.

[138] O. Vincent and M. Carlson. Sip4, a snf1 kinase-dependent transcriptional activator, binds to the carbon source-responsive element of gluconeogenic genes. *The EMBO journal*, 17(23):7002–7008, December 1998.

[139] Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10(1):57–63, January 2009.

[140] Werhli, V. Adriano, Grzegorczyk, Marco, Husmeier, and Dirk. Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and bayesian networks. *Bioinformatics*, 22(20):2523–2531, October 2006.

[141] J. Woolf, Prudhomme Wendy, Daheron Laurence, Q. Daley, and A. Lauffenburger. Bayesian analysis of signaling networks governing embryonic stem cell fate decisions. *Bioinformatics*, 21(6):741–753, March 2005.

[142] C. H. Yeang, Trey Ideker, and T. Jaakkola. Physical network models. *Journal of Comp. Biol.*, 11(2-3):243–246, 2004.

[143] Jing Yu, Anne A. Smith, Paul P. Wang, and Alexander J. Hartemink. Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594+, 2004.

[144] Kai Yu, John Lafferty, Shenghuo Zhu, and Yihong Gong. Large-scale collaborative prediction using a nonparametric random effects model. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1185–1192, New York, NY, USA, 2009. ACM.

[145] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 1012–1019, New York, NY, USA, 2005. ACM.

[146] Bai Zhang, Huai Li, Rebecca B. Riggins, Ming Zhan, Jianhua Xuan, Zhen Zhang, Eric P. Hoffman, Robert Clarke, and Yue Wang. Differential dependency network analysis to identify condition-specific topological changes in biological networks. *Bioinformatics*, pages btn660+, December 2008.