Methods for Reconstructing Networks with Incomplete Information

by

James Bruce Henderson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Statistics)
in The University of Michigan
2015

for Megan

ACKNOWLEDGEMENTS

First I'd like to thank my advisor George Michailidis for his collaboration on the work collected in this dissertation. It was George who first drew my attention to the class of network reconstruction problems and their role in the biological sciences. I remember that first meeting three years ago and the feeling of enthusiasm I felt for the many interesting questions that lay ahead. In the years since I've benefitted tremendously from his experience, breadth of knowledge, and ability to draw connections between different bodies of literature. As an advisor he struck a balance between providing structure and allowing independence that provided the environment for me to develop the confidence and perseverance needed for research.

I'd also like to thank the many excellent teachers and mentors I've had throughout my life. Thanks especially to the teachers and staff from Otsego Public Schools who kept me challenged and on track as well as to professors Jonathan Hall at Michigan State and Jeffrey Adler at American for reviving my interest in mathematics at times when it was waning. Jason Merrill and the entire Russian Department at Michigan State were instrumental in my intellectual development. The Statistics Department here at Michigan is full of outstanding teachers from whom I've been fortunate to learn. Special thanks to Kerby Shedden and Long Nguyen who were my faculty mentors during the often challenging first two years and to Yves Atchade and Ed Ionides for helpful conversations. Thanks to Kerby also for introducing me to another mentor, Patrick Nelson, through whom I've met a number of interesting scientific collaborators.

Thanks to Brenda Gunderson for her indefatigable positivity and for helping me and countless others grow as educators. Likewise thanks to Judy for always having a kind word and a smile. Thanks also to her and everyone else who keep the Department running smoothly.

Thanks to my graduate student peers for creating a collegial environment. Special thanks to Chye, Josh, and Karen for your friendship and support. I'd also like to acknowledge all those over the years who've supported our union, the Graduate Employees Organization, and put in the time to ensure we're fairly compensated for the teaching and research we do.

I couldn't have done this without the love an support of my family; I won't name you all but you know who you are. Mom, Laura, Deb, thanks for always believing in me. My dad was my intellectual role model long before graduate school. Devin your friendship and drive are a constant inspiration to follow ideas to their fullest. Finally, to my partner Megan, thank you for everything; I'm incredibly fortunate to share my life with you and ever grateful for your patience and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

AUC-PR	Area Under the Precision Recall Curve
AUC-ROC	Area Under the Reciever Operator Characteristic
BIC	Bayesian Information Criterion
CPDAG	Completed Partially Directed Acyclic Graph
DAG	directed acyclic graph
DREAM	Dialogue on Reverse Engineering and Assessment Methodologies
GCV	Generalized Cross Validation
MCMC	Markov Chain Monte Carlo
NeRDS	Network Reconstruction via Dynamic Systems
ODE	Ordinary Differential Equation
TDR	True Discovery Rate

ABSTRACT

Methods for Reconstructing Networks with Incomplete Information

by

James Henderson

Chair: George Michailidis

Abstract

Network representations of complex systems are widespread and reconstructing unknown networks from data has been the subject of intensive research in both statistical and scientific communities more broadly. Two primary challenges in network reconstruction problems include having insufficient data to illuminate the full structure of the underlying network and the need to combine information from data of different types. With these challenges in mind, this thesis contributes methodology for network reconstruction problems in three main respects.

The first part of this thesis considers the problem of sequentially choosing interventions to discover the structure of unknown directed networks with a particular focus on learning a partial order over the nodes. In Bayesian networks, data are modeled as having a joint distribution in which the value of nodal variables depend on the values of their parent nodes. Data arising from interventions such as gene knockouts are then treated using graph mutilations and intervention calculus. However, our focus on learning partial orders leads to a new model for intervention data under which the values of nodal variables depend on the lengths of paths separating them from the intervention target rather than on parent sets. We take a Bayesian approach to estimation, present priors based on partial orders, and develop a novel Markov Chain Monte Carlo (MCMC) method for computing the posterior over the space of directed acyclic graphs. The novelty and utility of the MCMC approach come from designing proposals for the Metropolis algorithm that move locally among partial orders while independently sampling graphs conditional on the partial order. The resulting Markov Chains mix rapidly and are shown to be ergodic. Finally, an existing information-theoretic strategy for active structure learning is adapted to the partial-order setting and an efficient Monte Carlo procedure for estimating the resulting decision function is developed. The proposed methods are evaluated numerically in simulations and using benchmark datasets for network reconstruction.

We next study penalized likelihood methods using incomplete order information. The goal in this chapter is to make use of the sort of order information arising from intervention data and studied in the chapter on active learning. To make the notion of incomplete information precise we introduce and formally define incomplete partial orders. This chapter subsumes as a special case existing work on estimating directed acyclic graphs using penalized likelihoods when a total ordering of the nodes is known. This special case is shown to lie along an information lattice and the reconstruction performance of penalized likelihood methods is studied at different points along this lattice.

In the final chapter, we present a method for ranking the potential edges in a network using time-course data. The novelty of the method lies in the development of a nonparametric gradient-matching estimation procedure and a related summary statistic for measuring the strength of the relationship among the components of a dynamic system. In simulation studies we demonstrate that given sufficient signal moving from linear to additive approximations using our nonparametric gradient-matching procedure leads to improved rankings of potential edges.

CHAPTER I

Introduction

1.1 Network Models and Network Reconstruction

Network models are useful for encoding dependencies among variables in complex systems because they represent these dependencies at an appropriate level of abstraction [58]. For instance, gene regulatory networks are often used to describe the functional relationships among a set of genes. Formally a gene regulatory network can be viewed as a directed graph with nodes corresponding to genes and edges indicating protein-mediated regulatory influences such as promotion or inhibition of gene expression. Learning the structure of such networks from data is a challenging problem and has been subject to intensive research [55, 15].

A number of formalisms have been proposed for learning networks from data [47, 55, 16]. Among these are: conditional independence models or Bayesian networks, direct cause-effect methodologies, and regression-style formalisms including dynamic systems models based on ordinary differential equations. In all cases, for biological applications nodes correspond to biochemical entities and edges to relationships among them, but the meaning of an edge within the model depends on the mathematical formalism employed.

As mentioned above, conditional independence models or Bayesian networks are a popular model for directed networks [67, 41]. Such a network is formally represented as a directed acyclic graph (DAG) encoding conditional independence relationships in a set of random variables represented as nodes. Edges then correspond to statistical dependencies, while nodal variables not connected by an edge are conditionally independent given their parents; i.e. expression levels for two genes are conditionally independent given the expression levels of their direct regulators. As another example, in a Bayesian network for a metabolic process, the concentrations of any two metabolites might be independent given the concentrations of their direct precursors and any enzymes facilitating the formation reaction.

There are two drawbacks to conditional independence models both stemming from the fact that the network being estimated is represented as a DAG. First, being acyclic DAGs cannot accommodate logically relevant cycles such as feedback loops [55]. Second, the number of potential DAGs grows super-exponentially with the number of nodes in the network necessitating approximate search strategies for even moderately sized networks [80]. While the problem of cycles can be overcome with time-series data by considering Dynamic Bayesian Networks [40, 65], this exacerbates the search problem due to the computational complexity involved for evaluating each potential network structure [80].

Nevertheless, conditional independence models are particularly well-suited for learning from gene knockout experiments. Indeed, the Dialogue on Reverse Engineering and Assessment Methodologies (DREAM) competitions provide strong evidence that knockouts are the most informative data type for reconstructing network topologies in biological applications [53]; see also discussion in [79]. However, in practice a full suite of intervention experiments is unlikely to be available due to financial, technical or ethical limitations. For instance, when reconstructing gene regulatory networks, there are a number of reasons why certain knockouts may be infeasible including lethality to the organism or the fact that, in many cases, how to do a particular knockout is unknown. In other biological applications where the nodes do not represent genes, such as in metabolic pathways, there may be no logical equivalent to a knockout experiment as one cannot, say, fix the concentration of an intermediary metabolite to zero. Moreover, the most successful methods for knockout data infer direct cause-effect relationships without necessarily making use of the conditional independence formalism [93]. However such methods tend to have difficulty distinguishing direct ($i \rightarrow k$) and indirect regulation ($i \rightarrow j \rightarrow k$) [69]. Consequently, these methods perform well when the influence and adjacency matrices are similar, but performance falls off when the adjacency matrix is much sparser than the influence or disruption matrix; i.e. when the network contains many chains of length two or more.

Another common set of methods take a regression-based approach to separately estimate parents for each node in the network [64]. Methods in this vein are appealing for their conceptual simplicity, the ability to include cycles, and computational tractability. The latter is especially important—by estimating the parent sets for each node separately regression-style approaches decompose large and potentially infeasible network estimation problems into more manageable subproblems. As further discussed in chapter IV, methods of this type are often cast in the context of dynamic systems models. In such models the evolution of the nodal variables through time is described by a system of equations expressing the rate of change for one variable in terms of the values of the others.

1.2 Incomplete Information

The primary subject of this thesis is network reconstruction and each of the remaining chapters addresses a different topic within this general area. However, there is also a common theme of incomplete information running throughout, further tying the chapters together. This theme is most prominent in the second and third chapters; in the former, we deal with choosing informative interventions having learned an incomplete description of the network structure and in the latter formally define a specific notion of incomplete information. The fourth and final chapter presents a network reconstruction method for time-course data assumed to arise from an underlying dynamic system for which the functional form of the governing equations is unknown. The following paragraphs give a brief overview of each chapter and touch on how they relate to this theme. More detailed descriptions including reviews of relevant literature appear in the introductions to individual chapters.

1.2.1 Active Learning and Path-Length Models

An important class of methods for network reconstruction is based on inferring direct-cause effect relationships using intervention data [93, 69]. Such methods are appealing because they make limited assumptions about the data-generating process. On the other hand, as previously discussed such methods are generally unable to distinguish direct from indirect causation while also requiring a more or less complete set of interventions to reconstruct the entire network. We take up some of these concerns in the second chapter after presenting a model based on path-lengths for inferring direct cause-effect relationships. Accepting that such models are ill-suited to distinguish direct and indirect effects, we seek instead to infer the partial order induced by the precedence relations of the underlying DAG. This provides a slightly higher-level description of the network structures consistent with the data and serves as a useful starting point for estimating the edgeset using refined models. Because in practice a complete set of interventions may be beyond reach, we also present an active learning strategy for economically choosing new interventions to further elucidate this aspect of a network's structure.

The active learning strategy uses a Bayesian approach and provides context for the additional contributions of this chapter. First, new priors based on partial orders are introduced and demonstrated to differ substantially from a uniform prior over

DAGs. The focus on partial orders also motivates a novel proposal distribution for use within the Metropolis-Hastings algorithm that allows the posterior to be sampled efficiently without resort to advanced techniques for preventing local trapping. Finally, a highly parallel Monte Carlo procedure is developed for estimating the decision function around which the active learning approach is based.

1.2.2 Using Incomplete Order Information

The third chapter explores incorporating the information gleaned from an incomplete set of intervention experiments or other prior knowledge into penalized-likelihood estimates of the edges in conditional independence networks. The chapter begins by defining incomplete partial orders as a way to describe restrictions on the precedence relations in the estimated graphs. The information content of an incomplete partial order can be viewed as varying along three axes in discrete steps. We present a simple plot for visually comparing the location of incomplete partial orders along the resulting information lattice. Incomplete partial orders generalize the important special case where a linear ordering of the nodal variables is assumed.

Two methods for incorporating the information contained in incomplete partial orders into estimates based on the Bayesian Information Criterion (BIC) are presented and compared. The first uses a stochastic search over the space of partial orders consistent with the incomplete information while the second uses an approach akin to neighborhood selection [56]. An example is used to compare methods at different locations along the lattice of incomplete order information.

1.2.3 A Nonparametric Method for Time-Course Data

The final chapter presents a regression-style network reconstruction method for use with time-course data that is appropriate when the underlying mechanisms can be modeled as a dynamic system governed by a collection of coupled differential equations. Dynamic systems models are frequently used for reconstructing biological networks and most such approaches amount to performing model selection for linear models [64]. Indeed, one of the more popular network reconstruction tools among computational biologists uses linear differential equations with a familiar ℓ_1 penalty [5, 49]. The novelty of our approach and its connection to the theme of incomplete information comes from not assuming a parametric class for the underlying differential equations but instead developing an additive nonparametric model using splines.

The resulting network reconstruction method estimates the system dynamics using a nonparametric gradient-matching procedure pulling together ideas for estimating parameters in parametric Ordinary Differential Equation (ODE) models [32] and additive modeling techniques for random variables [74]. The fitted models express the rate of change in nodal variables as a sum of univariate functions for other system variables. As with other regression-style methods the goal in this chapter is not to infer a single network but rather to produce a ranking of potential edges. To produce a ranking from the estimated dynamics, we propose a modified L_2 norm of the estimated functions as a coupling metric for measuring the influence of one node on another.

Working with simulated data from computational models for metabolic pathways and gene regulatory networks, the quality of these rankings is compared to rankings obtained from linear ODE models of various types. Linear ODE models provide a useful reference point as they are often justified as first-order approximations to non-linear models in settings where a parametric form for the latter is either unavailable or too difficult to obtain estimates for. These simulations studies demonstrate that the flexibility offered by a nonparametric approach results in improved network reconstruction performance provided the underlying time-series have sufficient signal.

CHAPTER II

Path-Length Models for Active Structure Learning

2.1 Introduction

In this chapter, we consider the problem of sequentially choosing interventions when attempting to recover the structure of unknown directed networks. While recovering the set of edges from the underlying DAG is of ultimate interest, here our focus will be on learning a higher-level structural description. Specifically, we focus on learning which pairs of nodes are connected by at least one directed path. Formally, we want to learn a *partial order* over the nodes.

This focus on learning partial orders leads us to a model for intervention data under which the values of the variables represented as nodes depend on the lengths of paths separating them from the intervention target. Path-length models can be contrasted with Bayesian networks modeling data as having a joint distribution having a factorization in which the value of nodal variables depend only on the values of their parents. Such models use graph mutilations and intervention calculus to describe data arising from interventions such as gene knockouts. Whereas parent sets and thus edges are of primary interest in a Bayesian network, the path-length models presented here emphasize the presence or absence of connecting paths mirroring the shift in focus from edges to partial orders.

In the active learning setting, we have the opportunity to choose a limited set of interventions sequentially with subsequent choices taking advantage of the structural information gleaned from earlier interventions. Each intervention tells us something about the network structure, but at a given stage the data from a limited set of interventions will generally be consistent with many DAGs. Because of this, we naturally take a Bayesian approach to estimation. As part of the estimation approach we present prior distributions based on partial orders. We also develop a novel MCMC method for sampling the posterior distribution over the space of DAGs. More precisely, we design proposals for the Metropolis algorithm based on partial orders which allow the Markov chains over DAG to move locally among partial orders while independently sampling graphs conditional on the partial order. The resulting Markov chains are ergodic and mix rapidly compared to standard structure MCMC. The overall active learning strategy used to choose interventions at each stage is similar to an existing information-theoretic approach, but the decision function we use is specifically adapted to the partial-order setting. In addition, we present an efficient and highly parallel Monte Carlo procedure for its estimation.

2.1.1 Structure Learning with Intervention Data

In important applications such as gene regulatory networks, intervention data are readily attainable and have been demonstrated to be of greater utility relative to non-intervention data for *de novo* network reconstruction [53]. While techniques for performing interventions in such applications are well established they require a good deal of technical expertise making intervention data generally more expensive to obtain than data from settings in which the network is not specifically perturbed. This motivates a sequential approach for choosing interventions in order to make good use of limited budgets.

Use of intervention data is necessary since, except in special cases [68], causal models are identifiable only up to Markov equivalence classes from observational data alone [67]. Even with high-quality intervention data, structure learning is challenging because the complexity of the space of directed graphs is super-exponential in the number of nodes. This makes it difficult to find 'good' models as determined by the likelihood or posterior distribution. In a Bayesian setting this difficulty is exacerbated by the need to average over all graphs to compute posterior probabilities of arbitrary features. While this intractable summation can be addressed using MCMC [51, 18, 30], existing methods are practically applicable only for small networks. Even for networks with tens of nodes, existing MCMC methods tend to suffer from slow-mixing with chains starting from different initial graphs getting stuck exploring small areas around local maxima in the posterior distribution. This has been attributed to both the larger number of samples needed to explore a parameter space with super-exponential complexity and the lack of smoothness in the posterior landscape owing to the discrete nature of graphs [23].

One approach to handling the challenge of slow-mixing is order-MCMC due to Friedman and Koller [23]. Drawing on optimization ideas from [9], order-MCMC constructs a Markov chain over linear orders rather than DAGs. This is done using the Metropolis-Hastings algorithm [36] and a trick that allows fast computation of likelihoods for linear orders assuming a bound on the maximum in-degree of each node. Friedman and Koller show that certain posterior features can be directly estimated from order-MCMC samples and suggest estimating other features by sampling DAGs from each order. In the latter case, order-MCMC introduces a bias because DAGs with fewer edges are consistent with more orders and consequently overrepresented in the sample. In theory, the bias can be corrected by counting the linear orders consistent with each DAG [19]. Unfortunately, counting linear extensions is #P-complete [6] though Niinimäki and Koivisto present a fast algorithm for doing so in networks with up to about 40 nodes provided the number of ideals is relatively small [62]. Another bias-correction approach studied by Ellis and Wong relies on sampling

DAGs from each linear order until $(1-\epsilon)$ of the conditional posterior is accounted for and then weighting accordingly [19].

Order-MCMC is quite useful but far from a panacea. For one, it requires using order-modular priors which have been criticized for distorting the resulting posterior [30]. Moreover, it is limited to the specific context of Bayesian Networks. This is undoubtedly a very large class of models, but far from the only context in which one may wish to compute a posterior over DAGs. For instance, in this chapter we directly model the effect of interventions based on path lengths instead of first setting up a joint distribution and then treating interventions using graph mutilations [67]. Such models do not admit a fast computation of the likelihood for a linear order. Nevertheless, the general idea of order-MCMC—working in a smaller space with smoother posterior landscape—can be adapted to construct improved proposal distributions for structure-MCMC as described in section 2.3.

In section 2.3 we present a new approach to structure MCMC based on *partial-order proposals* for use within the standard Metropolis-Hastings algorithm. The basic idea is to move locally among partial orders while independently sampling DAGs consistent with each partial order. The local moves through the space of partial orders resemble standard structure MCMC and allow the resulting chains to move stochastically toward regions of high posterior probability in DAG-space while drawing DAGs independently from each partial order helps to avoid local trapping and improve mixing. Linear orders are not suitable for such an approach as they do not form a partition of DAG-space as partial orders do. As also shown in section 2.3, the resulting chains are ergodic and have the correct stationary distribution.

2.1.2 Active Structure Learning

Faced with limited budgets, an important question in network reconstruction problems is how to systematically choose which experiments to carry out [43, 85, 83, 39, 38, 59]. This chapter describes an approach for active learning of network structure from intervention data such as ‘knockouts’ from gene-deletion mutants, ‘knockdowns’ via RNA-interference or other experimental techniques targeting a single node at a time. By active learning we mean that data are obtained sequentially with the experimenter able to choose at each step from among a set of available interventions. We build on the decision-theoretic approach for active structure learning in Bayesian Networks of Tong and Koller [85]; for a slightly different but related approach see also [60]. While using a similar framework for sequential decision making, we forgo Bayesian Networks in favor of a likelihood directly modeling interventions in terms of path-lengths.

Active learning algorithms for choosing interventions to orient edges in an undirected network estimated from observational data were studied by He and Deng [39] and later Hauser and Bühlman [38]. Our approach differs in that we do not begin with this undirected skeleton, focusing instead on uncovering precedence relationships—that is, learning a *partial order* on the nodal variables. In addition, our approach accounts for structural uncertainty after each stage of data collection whereas the methods above rely on a point estimate of the Markov equivalence class without regard to its uncertainty. Interesting approaches to sequentially choosing interventions in other classes of models include [59, 83].

The focus on partial orders is motivated in part by the observation that the set of potential interventions may be limited in practice due to the technical difficulty or lethality of some interventions. With limited data there may be many DAGs that describe the data well, making the presence or absence of particular edges sensitive to the specific form of the likelihood. However, due to the inherent directionality of intervention experiments the high-likelihood DAGs will often share a common set of precedence relationships making inference for partial orders more robust to model misspecification.

2.1.3 Summary

The focus in this chapter is on actively learning the partial order induced by the precedence relationships within a network represented as a directed graph. Our approach is nominally Bayesian in that we begin with a prior over the space of DAGs and that our loss function for selecting subsequent interventions is an expectation with respect to the posterior distribution. As in Tong and Koller, our loss function is a marginal posterior expected entropy though for a different set of marginals. Specifically, we replace marginal probabilities of edges with marginal probabilities of precedence relations so that the focus is on learning a partial order rather than a DAG. We also introduce novel priors based on partial orders.

The remainder of this chapter is structured as follows. In section 2.2, we present our model, fix notation, and outline an approach to actively choosing interventions. In section 2.3 we present a novel MCMC algorithm for estimating posterior expectations over graphs that is tailored to the path-length models used here. The approach is built around a new proposal distribution based on partial orders that allow the resulting Markov chains to mix rapidly. Sections 2.5 and 2.6, respectively, contain numeric work demonstrating the utility of the proposed techniques and applications on benchmark datasets. Finally, in section 2.7 we summarize our findings and point to interesting avenues for further research.

2.2 Path-Length Model and Active Learning Framework

2.2.1 Notation and Mathematical Background

Let $G = (V, E)$ be a DAG with $|V| = d$ nodes and denote the number of node pairs $s = d(d-1)/2$ as our attention will often be focused on pairwise relations. A (strict) *partial order* on V is a relation $\prec$ that is irreflexive, antisymmetric, and transitive. There is a unique partial order on V induced by G, $\Pi_G = \{(i,j) \in$

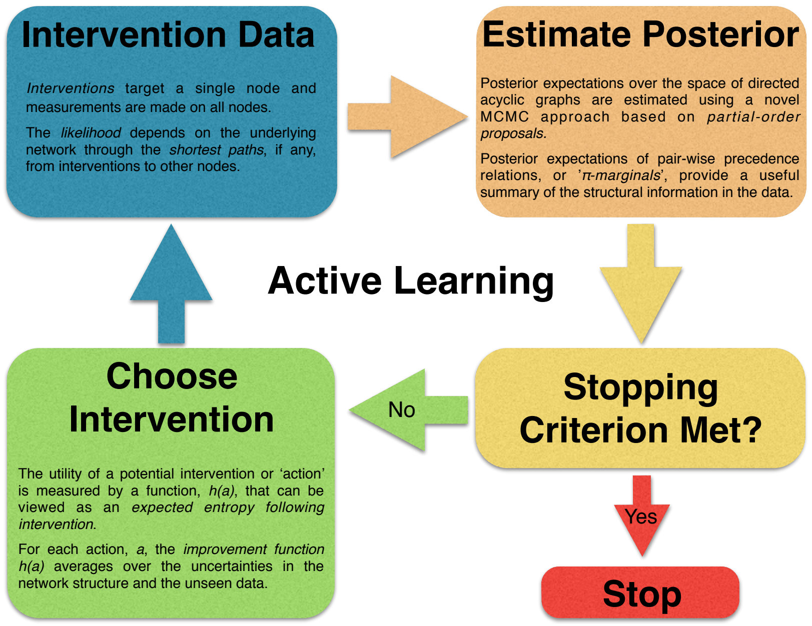

Intervention Data
Interventions target a single node and measurements are made on all nodes.
The *likelihood* depends on the underlying network through the *shortest paths*, if any, from interventions to other nodes.
Estimate Posterior
Posterior expectations over the space of directed acyclic graphs are estimated using a novel MCMC approach based on *partial-order proposals*.
Posterior expectations of pair-wise precedence relations, or '*π-marginals*', provide a useful summary of the structural information in the data.
Active Learning
Choose Intervention
The utility of a potential intervention or 'action' is measured by a function, *h(a)*, that can be viewed as an *expected entropy following intervention*.
For each action, *a*, the *improvement function h(a)* averages over the uncertainties in the network structure and the unseen data.
No
Stopping Criterion Met?
Yes
Stop

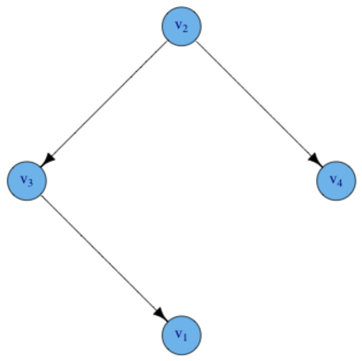
Figure 2.1: *Schematic overview of the active learning framework.* This section presents a model for intervention data based on path-lengths and also outlines an approach to active structure learning.

V^2 : there is a path from i to j in $G\}$. When $(i,j) \in \Pi$ write $i \prec_\Pi j$ and if neither $(i,j),(j,i) \in \Pi$ write $i \, ||_\Pi \, j$. For all $i,j \in V$ we define γ_i^j as the length of the *shortest* path from j to i in G. When no such paths exist, i.e. $(j,i) \notin \Pi_G$, set $\gamma_i^j = 0$. Note that $\gamma_i^j > 0$ implies $\gamma_j^i = 0$. Partial orders Π and DAGs G will often be denoted in pairwise formats, respectively $\pi = (\pi_k)_{k=1}^s$, and $\gamma = (\gamma_k)_{k=1}^s$, with components,

$$\pi_{\xi(i,j)} = \begin{cases} -1, & i < j, i \prec_\Pi j \\ 0, & i \not\prec_\Pi j, j \not\prec_\Pi i, \\ 1, & i < j, j \prec_\Pi i \end{cases} \qquad \gamma_{\xi(i,j)} = \begin{cases} -\gamma_j^i, & i < j, i \prec_{\Pi(\gamma)} j \\ 0, & i \not\prec_\pi j, j \not\prec_{\Pi(\gamma)} i \\ \gamma_i^j, & i < j, j \prec_{\Pi(\gamma)} i. \end{cases} \tag{2.1}$$

where $\xi : V^2 \to \{1,...,s\}$ is an indexing (say lexicographical) of the node pairs. See FIGURE 2.2 for an illustration on a toy example.

Functions $\Pi(\gamma)$ (with range a set) or $\pi(\gamma)$ (with range a vector) indicate the unique



i	j	$\xi(i,j)$	$\pi_{\xi(i,j)}$	$\gamma_{\xi(i,j)}$	γ_i^j	γ_j^i
1	2	1	-1	-2	2	0
1	3	2	-1	-1	1	0
1	4	3	0	0	0	0
2	3	4	1	1	0	1
2	4	5	1	1	0	1
3	4	6	0	0	0	0

Figure 2.2: *Notation for a toy graph.* On the left is a simple directed acyclic graph (DAG) used here for illustrative purposes. The nodes labels $v_1, \ldots, v_4$ are given the natural canonical ordering. On the right is a table giving various pairwise notations for this DAG. The first two columns list node pairs and the third shows ξ which orders the pairs, in this case lexicographically. *Partial Order:* The fourth column shows the partial order in pairwise format. In the first row, $\pi_{\xi(1,2)} = -1$ because 2 precedes 1 in the graph but $1 < 2$ in the canonical ordering. Similarly, in row 5 $\pi_{\xi(2,4)} = 1$ because 2 precedes 4 in the graph and $2 < 4$ in the canonical ordering. Finally in row 3, $\pi_{\xi(1,4)} = 0$ because no directed path exists between 1 and 4 in either direction. *DAG:* The fifth column gives the pairwise representation of the DAG. The signs {-, 0, +} follow the same rules as for the partial order. In row 1 the magnitude is different because the shortest path from node 2 to node 1 contains two edges. The final two columns give the length of the shortest path beginning at the node indicated in the superscript and terminating at the node indicated in the subscript.

partial order associated with DAG γ. Using this notation a positive $\gamma_{\xi(i,j)}$ gives the length of the shortest path from the smaller of i, j to the larger while a negative $\gamma_{\xi(i,j)}$ indicates the path goes in the reverse direction, i.e. from the larger of i, j to the smaller. The function $i(\gamma) \in \{-1, 0, 1\}^s$ gives the location and direction of edges in the corresponding DAG.

Next we provide notation for important sets. Let $\mathcal{L}$ be the set of all linear orders on V. A *linear order* is a partial order in which all pairs are comparable and will be represented as a permutation of the node labels, i.e. $L = \sigma(V)$. For $L \in \mathcal{L}$, $\Pi_L = \{\Pi : L_i < L_j \ \forall(i, j) \in \Pi\}$ is the set of all partial orders consistent with L. Similarly $\mathcal{L}_\Pi = \{L \in \mathcal{L} : L_i < L_j \ \forall(i, j) \in \Pi\}$ is the set of linear extensions for Π also denoted $\mathcal{L}_\pi$. Likewise $\Gamma_\pi = \{\gamma : i(\gamma) = \pi\}$ is the set of DAGs consistent with partial order π.

Finally, we introduce the *transitive reduction* $\check{\Pi}$ $(\check{\pi})$ of partial order Π $(\check{\pi})$ which plays an important role in the MCMC approach developed in section 2.3. The transitive reduction is formed by removing all relations $(i, j) \in \Pi$ for which there exists a mediating path $m_1, ..., m_\ell \in V$ with $(i, m_1), (m_1, m_2), \ldots, (m_\ell, j) \in \Pi$. For our purposes, it is best viewed as the DAG in Γ_π with the fewest possible edges; formed by removing all feedforward edges from any DAG in this class. For example, consider the DAG in Figure 2.2 which has edge set $E = \{(v_2, v_3), (V_2, v_4), (v_3, v_1)\}$. The partial order induced by precedence relations is $\Pi = E \cup \{(v_2, v_1)\}$ and the additional relation (v_2, v_1) is a feedforward motif since it is mediated by the path $v_2v_3v_1$. Since this is the only relation mediated by a path with more than two nodes, removing if from Π gives the transitive reduction $\check{\Pi} = \{(v_2, v_3), (V_2, v_4), (v_3, v_1)\}$; i.e. $\check{\Pi} = E$.

2.2.2 Model

Next we describe a model for the observed effects Y^α under intervention α. To fix ideas, assume that interventions set the perturbed variable to zero much like a gene knockout. The observed effects are assumed to lie in $[0,1]$ and follow truncated normal distributions with means $\{\mu_i^\alpha\}$ and known variance σ^2. Each intervention mean μ_i^α is a function of a random sign ψ_i^α determining the direction of the effect and a scale factor β_i^α determining its relative position between a baseline value μ_i^0 and the appropriate extremum. See Figure 2.3 for a graphical representation of the model, represented symbolically below:

$$\begin{aligned}
\gamma &\sim p(\gamma) \\
\{\beta_i^\alpha | \gamma_i^\alpha\} &\overset{ind}{\sim} \text{Beta}(2, 2 + c\gamma_i^\alpha)1[\gamma_i^\alpha \neq 0] \\
\{\psi_i^\alpha\} &\overset{iid}{\sim} \text{Binomial}(.5) \\
\mu_i^\alpha &= (\mu_i^0 - \mu_i^0\beta_i^\alpha)1[\psi_i^\alpha = 1] + (\mu_i^0 + (1-\mu_i^0)\beta_i^\alpha)1[\psi_i^\alpha = -1] \\
Y_{ik}^\alpha | \mu &\overset{ind}{\sim} \text{TN}(\mu_i^\alpha, \sigma^2; 0, 1); i = 1, ..., d; k = 1, ..., n.
\end{aligned} \tag{2.2}$$

Under this model, the relative magnitudes of indirect effects are stochastically smaller than direct effects and also decrease as the number of mediators in the shortest path grows. This allows for a situation frequently occurring in applications where effects diminish as they propagate through the network due to redundancies and (unmodeled) feedback mechanisms. The constant c allows for adjusting the rate at which effect sizes diminish; in this work we take $c = 1$. Finally, we remark that without this shrinking the likelihood would depend on γ only through the induced partial order.

Unlike methods for structure learning based on structural or differential equation models, we do not attempt to use the information content from the signs and magnitudes of effects. Instead we take them to be independent in order to make likelihood computations as efficient as possible. In fact, the signs and magnitudes will be marginalized out altogether. In effect we are trading inferential efficiency for computational efficiency while also guarding against model misspecification. Though not pursued here, the signs could be retained with the inferential target becoming

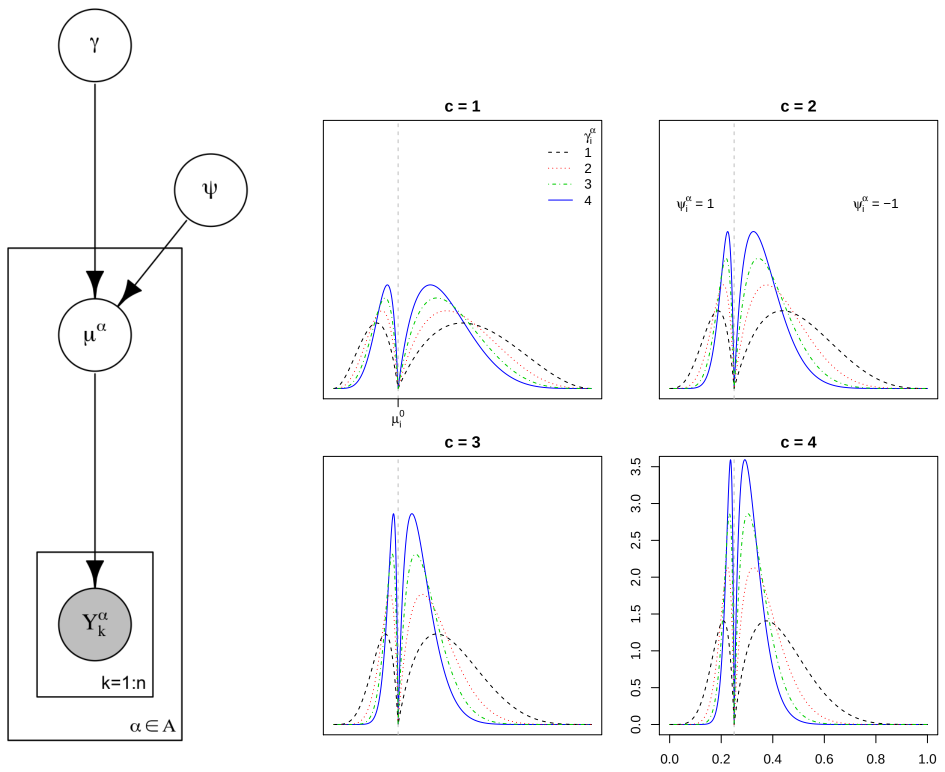


Figure 2.3: *Graphical representation of the intervention model (Left).* For each intervention α, a set of intervention means μ^α are determined by the DAG γ and signs ψ. The data Y_k^α are noisy observations of μ^α, both d-dimensional vectors. *Selected prior distributions for μ_i^α (Right).* Conditional on signs ψ^α, components of μ^α are given scaled and shifted Beta priors. The scaling and shifting are relative to non-intervention, i.e. wild-type, means μ^0, assumed known. The transformed Beta random variables are taken to be independent given γ and place more mass near μ_i^0 as the path-length γ_i^α grows. Each panel on the right shows the prior for minimum path-lengths 1-4 for different values of a parameter c, controlling how quickly the prior concentrates near the μ_i^0.

a DAG with signed edges. The same partial order framework would apply, but the number of DAGs consistent with each would grow from $2^{||\pi-\tilde{\pi}||_0}$ to $3^{||\pi-\tilde{\pi}||_0}$.

2.2.3 Prior Distributions

The standard non-informative prior is uniform over the space of DAGs, $p(\gamma) \propto 1$. While having the benefit of simplicity, this prior has the effect of giving larger weight to partial orders containing more relations since these are consistent with a greater number of DAGs. An alternative is to proscribe a prior that is uniform over partial

	$p(\gamma) \propto 1$			$p(\pi) \propto 1$		
d	-1	0	1	-1	0	1
10	0.44	0.12	0.44	0.25	0.50	0.25
15	0.46	0.08	0.46	0.23	0.54	0.23
20	0.47	0.06	0.47	0.22	0.57	0.22
25	0.48	0.05	0.48	0.21	0.59	0.21
30	0.48	0.04	0.48	0.20	0.60	0.20
50	0.49	0.03	0.49	0.20	0.60	0.20

Table 2.1: *Comparing priors uniform over DAGs to priors uniform over partial orders.* This table compares the π-marginals under two different non-informative priors for networks of various dimension d, corresponding to the number of nodes. Values on the left correspond to a prior uniform on DAGs and values on the right to a prior uniform on partial orders. Due to symmetry the prior π-marginals are the same for all pairs $\nu_i, \nu_j \in V$. For $i < j$, the column labels $-1, 0$, and 1, correspond to $\nu_j \prec_\pi \nu_i$, $\nu_i||_\pi\nu_j$, and $\nu_i \prec_\pi \nu_j$, respectively. With a prior uniform on DAGs a majority of mass is on the existence of a precedence relation and this majority grows with the dimension d. In contrast, under a prior uniform on partial orders the majority of mass is placed on no precedence relation and this majority also increases with d. Values may not add to 1 due to rounding.

orders. A simple choice within this class is $p(\gamma) \propto |\Gamma_{\pi(\gamma)}|^{-1}$ giving DAGs prior probabilities that are uniform conditional on the associated partial order. That this is a valid probability follows from the fact that partial orders partition the space of DAGs. These priors are compared in TABLE 2.1. Values in the table were estimated using 10 chains with $1e5$ samples each following $1e3$ burn-in samples using the MCMC procedure described in section 2.3. Relying on symmetry, for any pair of nodes (i, j), the probabilities $p(\pi_{\xi(i,j)} = -1)$ and $p(\pi_{\xi(i,j)} = 1)$ are estimated jointly and the reported values are averaged across both chains and pairs. More generally, conditional on the partial order one may prefer to give more weight to DAGs with fewer edges. For any real-valued monotone function s this is achieved by using a prior of the form,

$$p(\gamma) \propto \frac{s(||i(\gamma) - \check{\pi}(\gamma)||_0)}{|\Gamma_{\pi(\gamma)}| \sum_{\gamma' \in \Gamma_{\pi(\gamma)}} s(||i(\gamma') - \check{\pi}(\gamma)||_0)}. \tag{2.3}$$

A prior as in equation (2.3) can be seen as a formalization of the down-ranking procedure used by the top competitor in the DREAM 4 competition [69]. An advantage of this form of sparsity over priors which directly penalize the number of edges is that we avoid the odd situation in which the highest prior probability is placed on the empty network. A disadvantage is the need to compute the normalizing constant $\sum_{\gamma' \in \Gamma_\pi} s(||i(\gamma') - \check{\pi}||_0)$. The sum is potentially intractable due to the large size of $|\Gamma_\pi|$ for partial orders with many relations, but can be computed efficiently by exploiting the form of the summand. Letting $R = ||\pi - \check{\pi}||_0$ be the maximum number of 'optional' (feedforward) edges we can compute the normalizing constant efficiently using,

$$\sum_{\gamma' \in \Gamma_\pi} s(||i(\gamma') - \check{\pi}||_0) = \sum_{r=0}^{R} \binom{R}{r} s(r). \tag{2.4}$$

2.2.4 Active Learning Framework

The active learning framework described below is close to that of Tong and Koller [85]. Briefly, we consider a sequence of intervention experiments $(\mathcal{E}_t)_t$ with $\mathcal{E}_t = (\alpha, (Y^\alpha_{i,k})^{d,n}_{i=1,k=1})$ consisting of an intervention α and observations of the induced effects. For simplicity, assume the number of technical replicates n to be fixed and let the space of potential interventions be $\mathcal{A} = (1, \ldots, d)$. For step t, denote the sequence of observed interventions by $A_t = (\alpha_1, \ldots, \alpha_t)$ and define the *action space* $\mathcal{A}_t = \mathcal{A} \setminus A_t$ to consist of all remaining interventions.

We choose from among the potential actions $\mathcal{A}_t$ by finding an action that minimizes the entropy of the π-marginals,

$$H_\pi(Y^\alpha; \mathcal{D}) = -\sum_{\xi=1}^{s} \sum_{k=-1}^{1} p(\pi_\xi = k | Y^\alpha, \mathcal{D}) \log p(\pi_\xi = k | Y^\alpha, \mathcal{D}). \tag{2.5}$$

This differs from Tong and Koller who instead choose interventions to minimize the entropy of the posterior edge probabilities; a related approach would be to minimize the entropy of the posterior γ-marginals. As a function of yet-to-observe data Y^α, $H_\pi(\alpha; \mathcal{D})$ is not directly computable. Instead, we will use an estimate of its expectation,

$$h(\alpha) = \mathbb{E}[H(Y^\alpha, \mathcal{D})|\mathcal{D}] = \mathbb{E}_\gamma[\mathbb{E}_{Y^\alpha|\gamma}[H(Y^\alpha, \mathcal{D})|\gamma]|\mathcal{D}], \tag{2.6}$$

averaging over both structural uncertainty and the distribution of Y^α.

The choice to target interventions toward learning the partial order rather than the graph itself is based on the following heuristics. First, we argue that single-intervention experiments are inherently more informative for learning partial order relations than for distinguishing direct and indirect effects. For the former, the model need only adequately separate signal from noise, while for the latter it must discern between competing explanations of the signal. Because of this, learning partial orders is more robust against model misspecification than directly learning DAGs. Second, the π-marginals are easier to estimate than the γ-marginals because the space of partial orders, though still of super-exponential complexity, is smaller than the space of DAGs and has a 'smoother' posterior landscape. For complexity comparisons see entries A001035, for partial orders, and A003024 for DAGs in the Online Encyclopedia for Integer Squeneces [81]. Indeed, there has been work suggesting MCMC over partial orders improves mixing further than (linear) order-based sampling by cutting down on multi-modality [63]. Final justification comes from observing that with known partial order one can more efficiently estimate a DAG using a refined model. For instance, with known partial orders, exact structure discovery in a Bayesian Network can be done with time and space requirements of order $O(2^d)$ for moderately sized problems [66]. It is also possible to adapt methods utilizing known causal orderings to work with partial orderings, an idea developed further in chapter III [78].

2.3 MCMC for Sampling the Posterior

2.3.1 Posterior Distribution

The posterior probability of a DAG γ given data $\mathcal{D} = (y_i^{\alpha t})_{i,t}$ is $p(\gamma|\mathcal{D}) = p(\mathcal{D}|\gamma)p(\gamma)/p(\mathcal{D})$. As shown below, efficiently computing the likelihood $p(\mathcal{D}|\gamma)$ and prior $p(\gamma)$ is straightforward. The difficulty lies in obtaining the normalizing constant $p(\mathcal{D})$ due to the super-exponential complexity of $\mathcal{G}$. In this paper our primary use of the posterior distribution is computing posterior π- or γ-marginals, both taking the form of posterior expectations,

$$p(\pi_{\xi(i,j)} = k|\mathcal{D}) = \sum_{\gamma} 1[\pi_{\xi(i,j)}(\gamma) = k]p(\gamma|\mathcal{D}), \quad k = -1, 0, 1, \quad \text{and} \tag{2.7a}$$

$$p(\gamma_{\xi(i,j)} = k|\mathcal{D}) = \sum_{\gamma} 1[\gamma_{\xi(i,j)} = k]p(\gamma|\mathcal{D}), \quad k = -(d-1), ... -1, 0, 1, ..., (d-1). \tag{2.7b}$$

Observe that the posterior probability of an edge is given by the γ-marginal for $k = \pm 1$ with the sign depending on the position of the node indices in the canonical ordering.

2.3.2 Likelihood Computations

Under the model presented here, the observations $\mathcal{D} = (y_i^{\alpha t})_{i,t}$ are conditionally independent given γ. Thus, marginalizing over the sign and magnitude of effects, the likelihood factors as

$$p(\mathcal{D}|\gamma) = \prod_{t=1}^{T}\prod_{i\neq\alpha}\prod_{k=1}^{n} p(y_{ik}^{\alpha t}|\gamma) = \prod_{t=1}^{T}\prod_{i\neq\alpha}\int\prod_{k=1}^{n} p(y_{ik}^{\alpha}|\mu_i^{\alpha}(\beta,\psi))p(\beta|\gamma)p(\psi)d(\beta,\psi). \tag{2.8}$$

Crucially, for each intervention α the dependence on γ is only through the shortest path-length (if any) from node α to node i, i.e. $p(\beta_i^{\alpha}|\gamma) = p(\beta_i^{\alpha}|\gamma_{\xi(i,\alpha)})$, meaning that computing $p(y_i^{\alpha}|\gamma)$ does not require integration for each γ. Instead $d(d-1)$ integrations—one for each node at each possible level $|\gamma_{\xi(i,\alpha)}|$—can be precomputed and stored so that evaluating the log-likelihood $\log p(y_i^\alpha|\gamma)$ reduces to a simple summation of required terms.

2.3.3 MCMC with Partial-Order Proposals

2.3.3.1 Overview

Following standard practice, we use the Metropolis-Hastings algorithm to sample the posterior distribution and form Monte Carlo estimates of the posterior expectations (2.7a) and (2.7b). Briefly, given an initial DAG γ^0 and a proposal distribution $q(\gamma'|\gamma)$ a sample $\gamma^1 \cdots \gamma^N$ is obtained as follows:

```
for $n = 1, \dots, N$ do
    1. Sample $\gamma' \sim q(\cdot|\gamma^{n-1})$.
    2. Compute $\alpha = 1 \wedge \frac{p(\mathcal{D}|\gamma')p(\gamma')q(\gamma|\gamma')}{p(\mathcal{D}|\gamma)p(\gamma)q(\gamma'|\gamma)}$.
    3. With probability $\alpha$ set $\gamma^n = \gamma'$ otherwise $\gamma^n = \gamma^{n-1}$.
end for
```

The novelty of our approach lies in a specification of the proposal $q(\gamma'|\gamma)$ built on partial orders.

As discussed in the introduction, the standard proposal γ' is sampled uniformly from the neighborhood of γ consisting of all DAGs differing from γ by the presence or absence of a single edge. However, this approach quickly breaks down as the number of nodes increases with the resulting Markov Chains typically exploring a small region around a local maximum. This is often attributed to the topography of the posterior landscape in which high probability graphs are separated by regions of much lower probability. For observational data, proposals based on edge reversals help to alleviate this issue by an allowing an edge connecting two nodes with high marginal correlation to switch direction in a single step rather than by first being deleted and then added in the reverse direction [27, 30].

Edge reversals are useful because they are tailored to the conditional independence relations encoded in Bayesian Networks. Likewise, we construct an efficient proposal distribution by considering the structure of path-length models and the inherent directionality of intervention data. An important feature of the model presented here is that DAGs inducing the same partial order have similar likelihoods, hence similar posteriors. The reason for this is that, generally speaking, including or excluding a path has a greater effect on the likelihood than changing its length. Since partial orders partition the space of DAGs, another way to say this is that the likelihood tends to vary more between elements of the partition than within them.

2.3.3.2 Partial-Order Proposals

What follows is a high-level overview of a proposal distribution that takes advantage of the intimate relation between DAGs and partial orders. For further details, including pseudocode, see Appendix A. In essence each proposal consists of three steps: (1) deciding whether to resample the current partial order or propose a new one; (2) proposing a new partial order when required; and (3) sampling a DAG from the appropriate partial order. These steps constitute the basic algorithm below:

Algorithm 1 Partial-Order Proposals

```
1. With probability $\rho 2^{||\pi-\check{\pi}||_0-(d-1)(d-2)/2}$ set $\pi' = \pi$.
2. Otherwise, sample $\pi' \sim q_1(\cdot|\pi)$.
3. Sample $\gamma' \sim q_2(\cdot|\pi')$.
```

In this way the proposal distribution decomposes as $q(\gamma'|\gamma) = q_2(\gamma'|\pi')q_1(\pi'|\pi(\gamma))$. Sampling a DAG γ' from partial order π' is straightforward as it only requires choosing which feedforward edges to include. In particular, feedforward edges can be sampled independently. A useful default is to independently include 'optional' feedforward edges, in set notation $(i, j) \in \Pi/\check{\Pi}$, with probability $\frac{1}{2}$ making $q_2(\gamma'|\pi') = |\Gamma_{\pi'}|^{-1}$; i.e. uniform over the partial order. More sophisticated choices taking advantage of the data or informative priors are also possible.

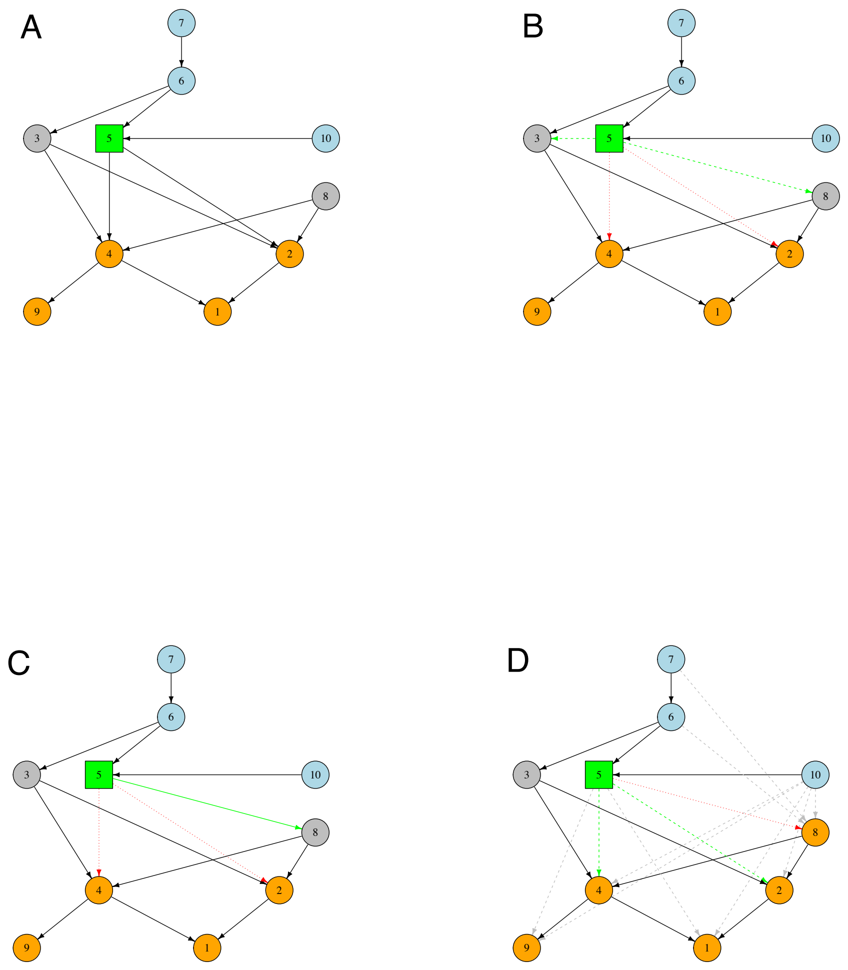


Figure 2.4: *Illustrations for partial-order proposals.* **A** A partial order on 10 nodes is shown here as its transitive reduction graph. The blue nodes are the ancestors of node '5' and the orange nodes are its descendants. The nodes in grey have no precedence relation to node '5'. **B** The legal moves involving edges emanating from node '5' begin by: adding edge $(5, 3)$, adding edge $(5, 8)$ removing edge $(5, 4)$, or removing edge $(5, 2)$. **C** Adding edge $(5, 8)$ causes edges $(5, 4)$ and $(5, 2)$ to become feedforward motifs which are removed when forming the transitive reduction. **D** The addition of $(5, 8)$ and subsequent removal of $(5, 4)$ and $(5, 2)$ is reversed by removing $(5, 8)$ and restoring $(5, 4)$ and $(5, 2)$. This reversal has nonzero probability since, when removing $(5, 8)$, all severed paths—those mediated solely by $(5, 8)$—have the potential to be included through systematic sampling.

The portion of $q(\gamma'|\gamma)$ corresponding to proposing a new partial order, $q_1(\pi'|\pi)$, is best understood as operating on the associated transitive reductions as illustrated in FIGURE 2.4. Viewing the transitive reduction, $\breve{\pi}$, as a DAG with no feedforward motifs, the proposed moves consist of familiar edge additions and deletions. Edge additions are only allowed between node pairs (i,j) such that $i \,||_\pi\, j$; in FIGURE 2.4B possible edge additions emanating from node '5' are shown in green and edges eligible for removal are colored red. When (i,j) is added to $\breve{\pi}$ we also remove from $\breve{\pi}$ all edges that become feedforward motifs after the addition as illustrated in FIGURE 2.4C. For addition of (i,j) to be reversible by the corresponding deletion there must be an opportunity for these removed edges to be restored concurrent to the deletion. This is accomplished by randomized decisions to include $(h,k) \in \breve{\pi}'$ when removal of (i,j) severs all paths in $\breve{\pi}$ from h to k. In FIGURE 2.4D all such pairs (h,k) are indicated by green and grey dashed lines with the green edges indicating the set that must be restored to return to the original partial order. These randomizations are done in a systematic order to ensure there is a unique sequence of inclusions leading from $\breve{\pi}$ to $\breve{\pi}'$ so that $q_1(\pi'|\pi)$ and $q_1(\pi|\pi')$ can be efficiently computed.

The probability with which a severed path (h,k) is included in $\breve{\pi}$ when sampled is given by a control parameter, $p \in (0,1)$, that can be adjusted to improve mixing. The other control parameter, ρ, is interpretable as the maximum probability of remaining in the current partial order. The probability for doing so, $\rho 2^{||\pi-\breve{\pi}||_0-(d-1)(d-2)/2}$, scales with the number of DAGs consistent with π relative to the maximum $2^{(d-1)(d-2)/2}$. Intuitively, the idea is to spend more time exploring larger partial orders. We have had success with $\rho = .9$ and $p = .5$ as defaults for these control parameters.

Though a detailed discussion is beyond the scope of this chapter, we note that the partial-order proposals described above are likely to be useful beyond the class of models considered here. For one, the scheme above can be used for sampling the posterior of a Bayesian Network though it remains to be seen if the improvements observed here will carry over. Furthermore, one can use the proposals above to develop *partial order MCMC* in a manner analogous to order MCMC as the fast computation for computing the posterior of a linear order based on limiting the in-degree of nodes can be directly adapted to a partial-order setting. This idea was previously pursued for a limited class of partial orders in [63].

2.3.3.3 Ergodicity

By construction, the Markov chains resulting from the procedure above are reversible, hence aperiodic, due to the correspondence between adding and deleting edges from the transitive reduction. Moreover, through use of the Metropolis-Hastings acceptance ratio they will satisfy detailed balance and have stationary distribution $p(\gamma|\mathcal{D})$ as desired. These facts constitute the following lemma.

Lemma 2.1. *The Markov Chains generated by* Algorithm *2 are reversible and have stationary distribution $p(\gamma|\mathcal{D})$.*

That this stationary distribution is also the limiting distribution follows from irreducibility which is clear given the non-zero probability of getting to the null partial order through a series of removals and from there to any other partial order through a series of additions. This is stated formally in the lemma below. See appendix A for formal proofs of this and the preceding lemma.

Lemma 2.2. *The Markov Chains generated by* Algorithm *2 are irreducible.*

Because the space of DAGs is finite, irreducibility implies all graphs are positive recurrent. The facts above are quite standard but sufficient to ensure the distribution of the process $\{\gamma_n\}_{n\geq 1}$ converges to $p(\gamma|\mathcal{D})$ in total variation and to provide a law of large numbers. The theorem below follows from direct application of Theorem 14.2.53 in [10]; see also Theorem 10.6 in [45].

Theorem 2.3. *Let $\{\gamma_n\}_{n\geq 1}$ be a sample obtained according to the Metropolis-Hasting algorithm above. Then, for any real-valued function $f : \Gamma \to \mathbb{R}$,*

$$\lim_{n\to\infty} n^{-1} \sum_{t=1}^{n} f(\gamma_t) = \sum_{\gamma\in\Gamma} f(\gamma)p(\gamma|\mathcal{D}) \ a.s..$$

2.4 Estimating the Entropy Under New Interventions

2.4.1 Decomposing the Improvement Function

Given data $\mathcal{D}$ from interventions $\alpha_{1:t}$ we wish to measure the utility of potential interventions $\alpha \in \mathcal{A}_t$ for refining our knowledge of the underlying network structure. In order to choose the next intervention $\alpha \in \mathcal{A}_t$, it is necessary to estimate the additional structural information contained in new data Y^α. This structural information is measured using the expected entropy of the π-marginals, $h(\alpha)$, defined in (2.5) and (2.6) and henceforth called the *improvement function*. The expectation in the definition of $h(\alpha)$ averages over both the structural uncertainty and the observational uncertainty given the structure. This can be seen by writing,

$$h(\alpha) = \mathbb{E}[H(Y^\alpha, \mathcal{D})|\mathcal{D}] = \mathbb{E}_\gamma[\mathbb{E}_{Y^\alpha|\gamma}[H(Y^\alpha, \mathcal{D})|\gamma]|\mathcal{D}]. \tag{2.9}$$

The decomposition above offers some insight into estimating $h(\alpha)$ by separating the inner and outer expectations.

Provided we can estimate $\mathbb{E}_{Y^\alpha|\gamma}[H(Y^\alpha, \mathcal{D})|\gamma]$, the outer expectation is easily estimated by sampling from the posterior $p(\gamma|\mathcal{D})$ as described in the previous section. In particular, we use a subsample drawn from the Monte Carlo sample used to estimate the posterior marginals given $\mathcal{D}$. The outer expectation can then be estimated using a simple sample average of the estimated inner conditional expectations. Estimates for the inner expectations given a DAG γ are described below.

2.4.2 Monte Carlo Estimate of the Inner Expectation

For fixed γ, we will use a Monte Carlo estimate of the inner expectation $\mathbb{E}[H(Y^\alpha, \mathcal{D})|\gamma]$. Specifically, we first simulate n_{mc} draws of $Y^\alpha|\gamma$ from the model (2.3) and use the estimate,

$$\hat{\mathbb{E}}_{Y^\alpha|\gamma}[H(Y^\alpha, \mathcal{D})|\gamma] = {n_{mc}}^{-1} \sum_{m=1}^{n_{mc}} \hat{H}(\mathcal{D}^\star_m) \tag{2.10}$$

with $\mathcal{D}^\star_m$ the data $\mathcal{D}$ augmented with the m^{th} simulated observation $Y^\alpha|\gamma$ and $\hat{H}$ defined below. Computing $\hat{H}(\mathcal{D}^\star)$ for each simulated Y^α requires estimating new posterior π-marginals, $\{\hat{p}(\pi_\xi = k|\mathcal{D}^\star)\}_{\xi,k}$ using the original Monte Carlo sample $(\gamma_n)_n$ and forming the estimate,

$$\hat{H}(\mathcal{D}^\star; (\gamma_n)_n) = -\sum_{\xi=1}^{s} \sum_{k=-1}^{1} \hat{p}(\pi_\xi = k|\mathcal{D}^\star; (\gamma_n)_n) \log \hat{p}(\pi_\xi = k|\mathcal{D}^\star; (\gamma_n)_n). \tag{2.11}$$

It should be pointed out that $\hat{H}$ has a positive bias as an estimator of H due to Jensen's inequality since $p \log p$ is concave in p. By the continuity of $p \log p$ it nevertheless remains consistent as the size of the inner Monte Carlo sample grows. While the bias may effect the action chosen, this action can be interpreted as minimizing an approximate upper bound on the expected entropy of the posterior π-marginals, $h(\alpha)$. If one is worried about the bias, its magnitude could always be estimated by computing $h(\alpha)$ using a sequence of increasingly larger Monte Carlo samples for estimating $\hat{H}$ and then applying a simulation-extrapolation-type procedure [14]. In practice, however, it is usually sufficient that the chosen action be among the best as opposed to optimal.

Drawing a new sample with stationary distribution $p(\gamma|Y^\alpha, \mathcal{D})$ is computationally infeasible since $H(Y^\alpha, \mathcal{D})$ needs to be estimated $|\mathcal{A}||\Gamma_{out}|n_{mc}$ times where Γ_{out} is the set of DAGs used for estimating the outer expectation. Instead, we use (a subset of) the original sample from $p(\gamma|\mathcal{D})$ as the instrumental distribution in a self-normalized importance sample [10],

$$\begin{aligned} \hat{p}(\pi_\xi = k|\mathcal{D}^\star; (\gamma_n)_n) &= \frac{\sum_{n=1}^N 1[\pi_\xi(\gamma_n) = k] p(Y^\alpha, \mathcal{D}|\gamma_n) p(\gamma_n)/p(\gamma_n|\mathcal{D})}{\sum_{n=1}^N p(Y^\alpha, \mathcal{D}|\gamma_n) p(\gamma_n)/p(\gamma_n|\mathcal{D})} \\ &= \frac{\sum_{n=1}^N 1[\pi_\xi(\gamma_n) = k] p(Y^\alpha|\gamma_n) p(\mathcal{D}|\gamma_n) p(\gamma_n)/(\gamma_n|\mathcal{D})}{\sum_{n=1}^N p(Y^\alpha|\gamma_n) p(\mathcal{D}|\gamma_n) p(\gamma_n)/p(\gamma_n|\mathcal{D})} \quad (2.12) \\ &= \frac{\sum_{n=1}^N 1[\pi_\xi(\gamma_n) = k] p(Y^\alpha|\gamma_n)}{\sum_{n=1}^N p(Y^\alpha|\gamma_n)}, \end{aligned}$$

for each Monte Carlo draw in the inner expectation. This should provide good estimates since the high-probability region of $p(\gamma|\mathcal{D}^\star)$ is largely contained in that of $p(\gamma|\mathcal{D})$. The size of the subset used is denoted N_{in}.

Finally, we remark that the inner and outer expectations need not—in fact should not—use the same subsamples. In particular, the reduced sample used to estimate the inner expectation can be much larger as the only additional computational burden is in the summation and indexing needed to compute updated likelihoods for correctly weighting DAGs when computing the simulated entropies. The size of these subsamples together with n_{mc} can be balanced against the size of the action space $|\mathcal{A}|$ and available computational resources when estimating the improvement function.

2.5 Numeric Work

This section presents selected simulation results demonstrating the utility of the above active learning framework as well as practical aspects of its implementation. While considering a variety of network structures, a common model will be used for simulating intervention data. Namely, the intervention means are modeled deterministically as

$$\mu_i^\alpha = \mu_i^0 + \sum_{j \in \mathrm{pa}_i} \beta(\mu_j^\alpha - \mu_j^0) \quad (2.13)$$

with $\mu_\alpha^\alpha = 0$ and pa_i the parents of node i. The observations follow a truncated normal distribution on $[0, 1]$,

$$Y_i^\alpha \sim TN(\mu_i^\alpha, \sigma^2; 0, 1). \tag{2.14}$$

Except where specified otherwise, all wild-type means are $\mu_i^0 = .5$, the noise variance is $\sigma = .05$, there are $n = 3$ technical replicates for each intervention and $\beta = .45$. While simulating from this model, computations are done under the model presented in section 2.2 which can be viewed as a pseudo-likelihood.

2.5.1 Improved Mixing using Partial-Order Proposals

This section is intended to briefly demonstrate the improvement offered by MCMC with partial-order proposals over classical structure MCMC. As previously discussed in section 2.1, classical structure MCMC is notorious for local trapping and slow mixing. As a result, it usually needs to be combined with computationally intensive MCMC techniques like simulated annealing or equi-energy sampling. Without these techniques situations like that in the bottom plot of Figure 2.5 are typical. On the bottom of Figure 2.5 are chains using standard structure MCMC with proposals based on adding or deleting a single edge. While rapidly leaving regions where the likelihood is very small the chains eventually get stuck exploring regions around different local maxima despite early excursions near the global maximum. In contrast, at the top of the figure are chains from the new approach which rapidly mix about this global maximum. By global maximum we are referring to the likelihood as there are potentially many DAGs with likelihoods at or near this maximum. Comparisons are for the 10-node Ecoli 1 network from the DREAM 3 competition after interventions at nodes 1,2,3, and 8 [70].

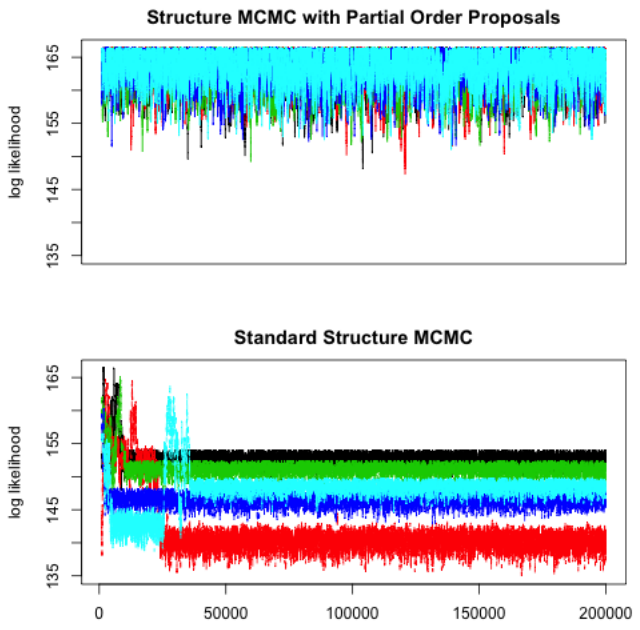


Figure 2.5: *MCMC with partial-order proposals improves mixing.* Each plot in this figure traces the (unnormalized) likelihood for five chains starting at random graphs. Shown here are steps 1,000 to 200,000. On bottom are chains using standard structure MCMC with proposals based on adding or deleting a single edge. On top are chains from MCMC with partial-order proposals starting from the same random graphs. While the former settle into local maxima away from the highest regions of the likelihood the latter mix well in this region.

2.5.2 Examples on 15- and 25-node Networks

In this section, we first illustrate the steps involved in the active learning framework on a synthetic 25-node network. We assume the stopping criterion is a fixed budget of ten interventions and that the initial intervention occurs at node '1'. The DAG to be reconstructed is shown in FIGURE 2.6. The figure also indicates the locations of the first ten interventions, the final nine selected by active learning, using square nodes with subscripts giving the timing of each intervention. Data is simulated according to (2.13) and (2.14) with $\mu_i^0 = .5$ for all nodes i, $\beta = .65$, $\sigma = .05$, and $n = 3$. An example of what the simulated data looks like following an intervention at node '3' appears as FIGURE 2.7.

Following each intervention, the first step is to estimate the posterior π-marginals using MCMC with partial-order proposals as discussed in section 2.3.3. In this example we use three chains, $\rho = .9$, $p = .5$, and draw $2e5$ samples discarding the first half. The chains are given warm starts as follows. After the initial intervention we compute z-scores $z_i = |\bar{Y}_i^{\alpha_1} - \mu_i^0|/\sigma$ for each node and include (α_1, i) in the initial partial order if $z_i > 2$. Following subsequent interventions chains are initialized using the final partial order sampled from a chain in the previous step. Convergence is assessed by examining trace plots of the likelihood, monitoring acceptance rates, and using the potential-scale reduction factor $\hat{R}$ [26]. For the latter values near one indicate convergence while numbers greater than one indicate the amount by which the Monte Carlo variance could be reduced by running the chains longer. In this case, after the third intervention the chains had acceptance rates of 17-18% and $\hat{R} = 1.01$ indicating good convergence. See FIGURE 2.8 for trace plots and TABLE 2.2 for the posterior π-marginals of 10 pairs selected at random from the 300 total pairs.

Following estimation of the posterior π-marginals, the MCMC sample is used to estimate the improvement function $h(\alpha)$ for all unseen interventions α. This is done using the Monte Carlo procedure described in section 2.4. In brief, from each of

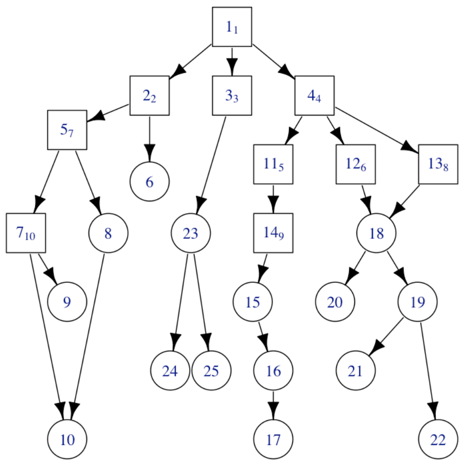


Figure 2.6: *Topology and first 10 interventions on a 25-node example.* This figure shows the (unknown) DAG to be reconstructed in the 25-node example. Square nodes indicate the locations of the first ten interventions as selected by the active learning framework given an initial intervention at node 1. The large-type numbers are labels for the nodes while the subscripts indicate the step at which each intervention was performed.

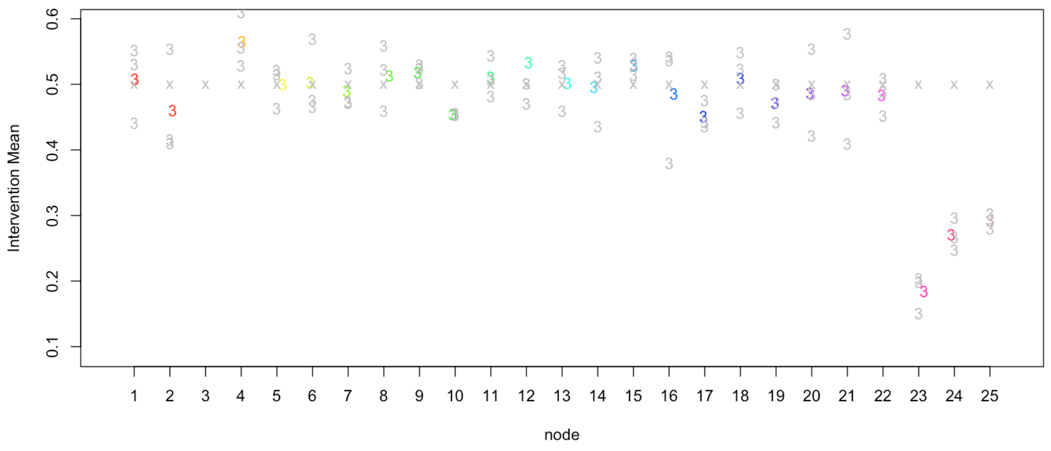


Figure 2.7: *Simulated data following intervention to node '3' in the 25-node example.* The wild-type means μ_i^0 are indicated with an 'x', grey numbers are individual observations, and values in color are the observed means.

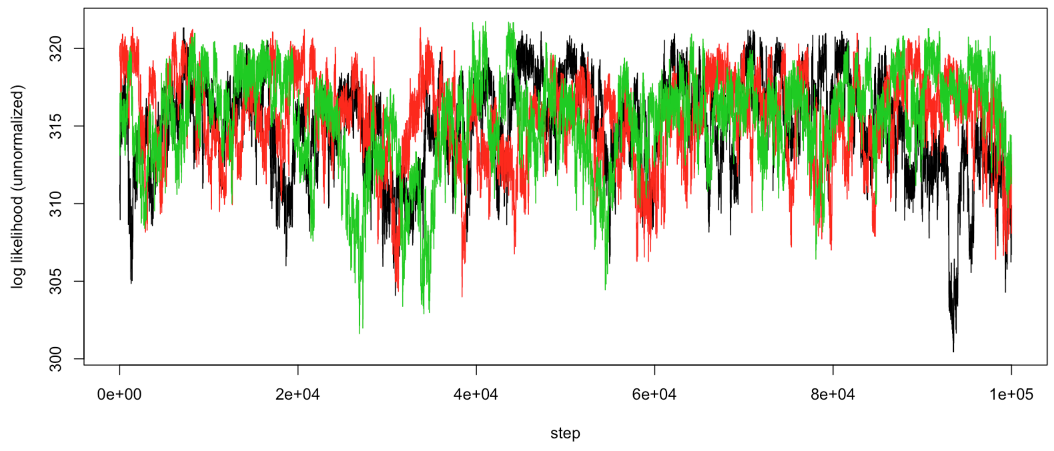


Figure 2.8: *Trace plots for three chains at step 3 on the 25-node example.* Only the 1e5 samples following the burn-in period are shown. The chains have acceptance rates from 17-18% and exhibit good mixing with $\hat{R} = 1.01$.

the three chains we select $N_{out} = 100$ DAGs for estimating the outer expectation and $N_{in} = 10,000$ DAGs for the inner expectation. For each DAG in the outer expectation and each action α we simulate $n_{mc} = 10$ Monte Carlo samples from the model (2.3). Note that this *is not the same* as the (unknown) data-generating model (2.13). For each Monte Carlo sample we estimate new posterior π-marginals using the N_{in} samples according to (2.13) and then estimate the entropies as in (2.11). These are combined using the sample average as in (2.10) and then averaged a final time over all DAGs in the outer expectation to estimate $h(\alpha)$. In this way, an estimate is obtained from each chain. Plots of these estimates as in FIGURE 2.9 can then be examined to either choose the next intervention or choose actions for which to obtain further Monte Carlo samples. In simulations the next action is simply chosen to minimize the average across chains.

There is no debating that the procedure for estimating the expected marginal entropies is computationally intensive. However, it admits a number of efficiencies the most important of which is being highly and trivially parallelizable. With almost no effort the procedure above can be parallelized over actions, chains, DAGs in the outer expectation and simulation draws. Given enough processors the expected entropies

i	j	$P(v_j \prec_\pi v_i \mid \mathcal{D})$	$P(v_i \mid\mid_\pi v_j \mid \mathcal{D})$	$P(v_i \prec_\pi v_j \mid \mathcal{D})$
2	4	0.03 (0.02)	0.64 (0.11)	0.33 (0.11)
4	15	0.25 (0.12)	0.59 (0.12)	0.16 (0.06)
6	22	0.48 (0.08)	0.50 (0.08)	0.02 (0.03)
9	15	0.36 (0.14)	0.64 (0.14)	0.00 (0.00)
10	11	0.40 (0.03)	0.59 (0.04)	0.01 (0.01)
13	22	0.26 (0.01)	0.63 (0.04)	0.11 (0.04)
16	20	0.02 (0.01)	0.75 (0.05)	0.24 (0.06)
16	21	0.01 (0.00)	0.58 (0.06)	0.41 (0.06)
17	25	0.01 (0.01)	0.39 (0.03)	0.60 (0.04)
22	24	0.00 (0.00)	0.38 (0.17)	0.62 (0.17)

Table 2.2: *Some posterior π-marginals for the 25-node example.* The table shows ten of the 300 posterior π-marginal following interventions to nodes '1', '2', and, '3'. Values are given as mean (standard deviation) from three chains. Ten pairs were selected at random for space considerations.

from this example could be computed in a manner of minutes with the primary bottleneck being collation of the many samples. In practice, we parallelize over actions and the outer expectation with chains and simulation draws run serially, the former outside the parallelization and the latter within. Using 16 cores with shared memory each chain in this example took a few hours. Further efficiency could be gleaned by grouping nodes that are close to interchangeable given the available information and considering actions as random draws from these groups. Another option would be to use an adaptive strategy in which actions are repeatedly separated into 'better' and 'worse' using smaller Monte Carlo samples with only the 'better' actions investigated in subsequent rounds.

As seen in Figure 2.9, for our example the next intervention is $\alpha_4 =$'4'. While this is intuitively appealing given that we know the DAG structure in the simulation setting, it should also be noted that an intervention at node '4' is not statistically distinguishable from interventions at nodes '11', '12', '13', '14', '16', '18', or '19' given the Monte Carlo variance. On the one hand this could be resolved by drawing additional Monte Carlo samples. On the other, the important thing to notice is that previous interventions at nodes '2' and '3' have created a loose partition of

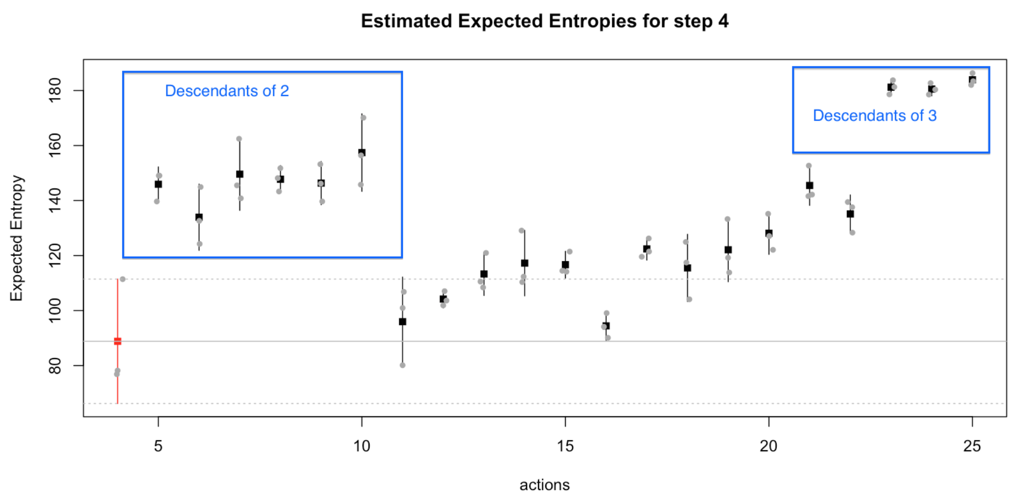


Figure 2.9: *Estimates of* $h(\alpha_4)$ *after step 3 of the 25-node example.*

the nodes into descendants of '2', descendants of '3', and descendants of neither. What the plotted entropies tell us is that it is best to choose the next intervention from among this last set containing the descendants of neither. When only a few interventions have been done, our knowledge of the underlying network structure is highly incomplete as there are many DAGs that describe the data well. However, even with incomplete information some actions will be better than others. Rather than chasing small differences in search of an optimal intervention, the real utility of an active learning approach lies in choosing from among the better actions.

The estimated entropy of the posterior π-marginals following each of the ten budgeted interventions is shown in FIGURE 2.10. It is compared to 100 random draws over the nine interventions to follow $\alpha_1 = 1$ shown as a box plot at step 10. Similar comparisons are made in FIGURE 2.11 for two 15-node networks extracted from the 50-node Ecoli 1 and Ecoli 2 networks of the DREAM 3 competition [70]. In these two examples all setting were as before except the signal was reduced from $\beta = .65$ to $\beta = .45$. These figures clearly demonstrate the utility of active learning which results in a lower marginal entropy after ten steps in all but a few cases. Indeed, it can be seen that active learning already matches or exceeds the median performance of ten

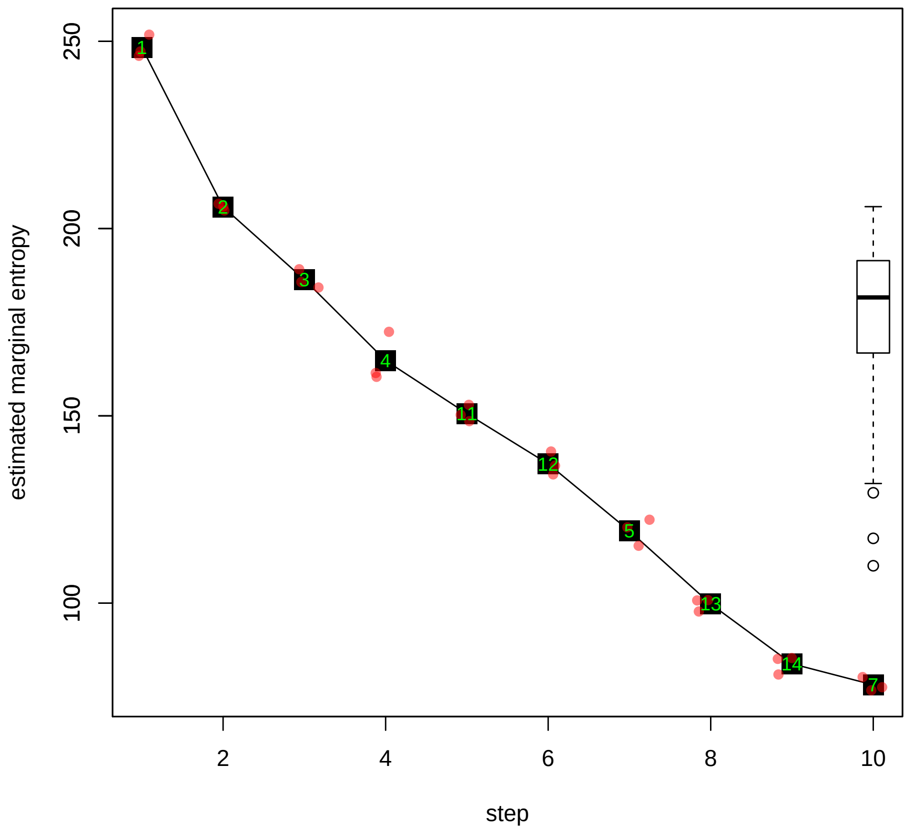


Figure 2.10: *Entropy after each step in the 25-node example.* The squares connected with a line show the entropy after each step. Numbers within squares indicate nodes at which interventions occurred. The box plot at step 10 gives the entropies from 100 random draws over interventions 2-10. The red circles at each step show the estimated marginal entropies from three separate Markov chains.

random interventions after either four or six interventions depending on the network.

2.5.3 Active Learning Performance on 10-node Networks

We conclude this section by investigating the performance of the active learning framework on the five 10-node networks from the DREAM 3 competition [70]. For each network we first generate 10 intervention data sets and then apply the active learning approach described in section (2.2) and illustrated in the previous subsection. The active learning approach is applied for selected initial interventions using a fixed

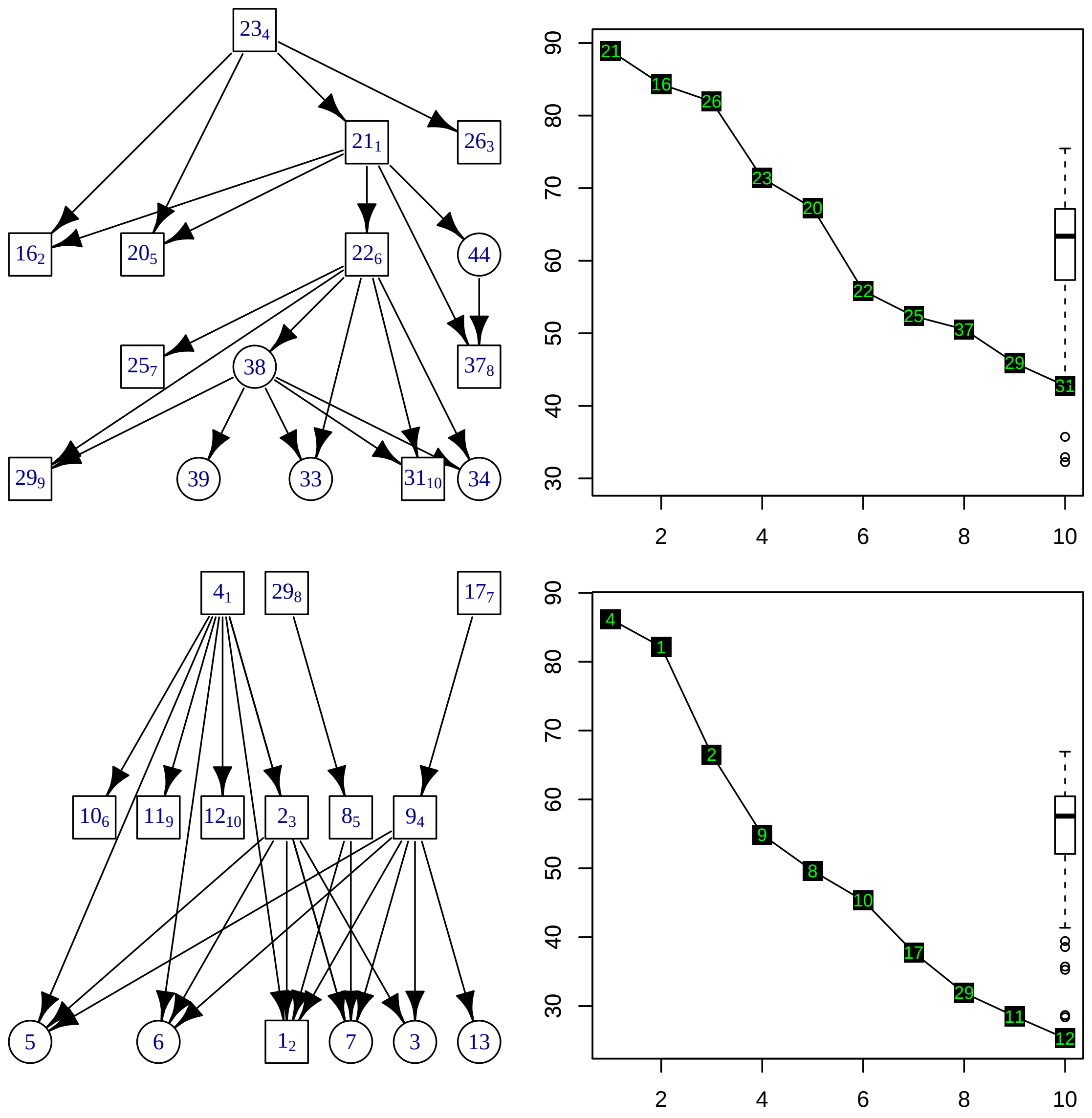


Figure 2.11: *Entropy comparisons in two 15-node examples.* On the left are two 15-node networks with squares indicating interventions under active learning and subscripts giving the timing of those interventions. On the right are the entropy paths for the particular sequence of interventions indicated. For comparison, box plots at step 10 gives the entropies from 100 random draws over interventions 2-10.

budget of 5 interventions as a stopping criterion. The entropy of the posterior π-marginals (see (2.7a)) after 5 interventions is used as an evaluation criterion. For each network and initial intervention the entropy of the active learning approach is compared to 100 entropies using randomly selected interventions, with 10 draws over $(\alpha_2, ..., \alpha_5)$ for each of the 10 data sets.

The results are compared using box plots in FIGURES 2.12, 2.13, 2.14, 2.15, 2.16, for the Ecoli 1, Ecoli 2, Yeast 1, Yeast 2, and Yeast 3 networks, respectively. For both cases the initial intervention is given by the label on the horizontal axis. For each initial action, α_1, the bar plots show how many times each possible intervention occurred among the first five across the 10 data sets and indicate that the interventions selected are fairly stable. All posteriors were estimated using MCMC with partial-order proposals with $1e5$ samples and a $1e5$ burn-in period for each of 3 chains. For the active learning approach interventions were chosen using $N_{out} = 100$, $N_{in} = 1e4$, and $n_{mc} = 10$.

Depending on the initial action, active learning is either significantly better or no worse on average than choosing interventions randomly. The initial interventions for which there is no clear improvement share the common trait of having all or most other nodes as descendants. Such interventions are valuable for uncovering descendant relations, but they tell us little about the possible relations between pairs of nodes that do not include the intervention target. Similar behavior would be expected for leaf nodes with no descendants, as neither type partitions the node set. However, whereas after an initial intervention on a leaf node the next intervention is more or less random, for an initial intervention on a source node the next intervention will often target the node with the largest perturbation. The lack of improvement in some cases can also be attributed in part to the relatively small number of interventions available as there is insufficient time for information to accumulate and make active learning beneficial.

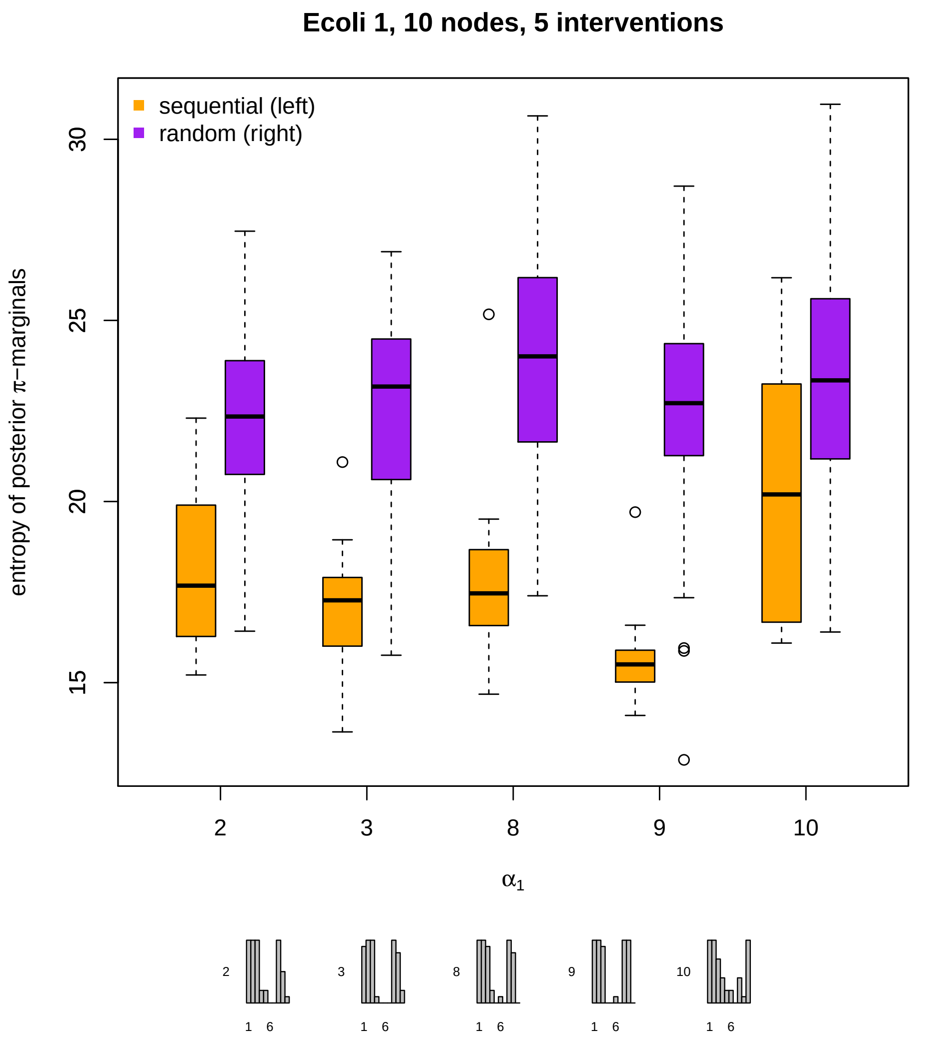


Figure 2.12: *Active learning performance at step 5 on Ecoli 1. Top:* Box plots comparing the entropy of the posterior π-marginals after five steps on the 10-node Ecoli 1 network. Each pair of box plots compare the active learning approach described in section 2.4 (right) to randomly chosen interventions (left). The label on the horizontal axis indicates the node targeted by the first intervention. *Bottom*: For each initial action, α_1, the bar plots show how many times each possible intervention occurred among the first five across 10 simulated data sets.

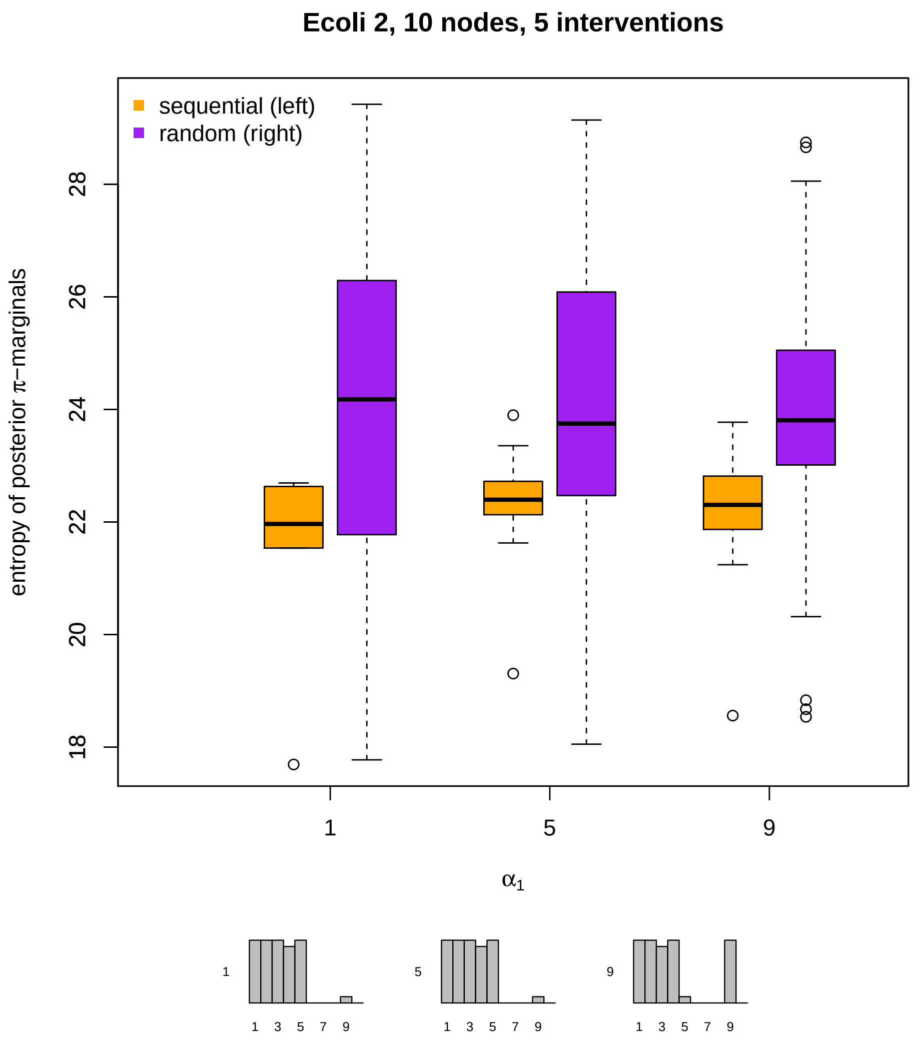


Figure 2.13: *Active learning performance at step 5 on Ecoli 2. Top:* Box plots comparing the entropy of the posterior π-marginals after five steps on the 10-node Ecoli 2 network. Each pair of box plots compare the active learning approach described in section 2.4 (right) to randomly chosen interventions (left). The label on the horizontal axis indicates the node targeted by the first intervention. *Bottom*: For each initial action, α_1, the bar plots show how many times each possible intervention occurred among the first five across 10 simulated data sets.

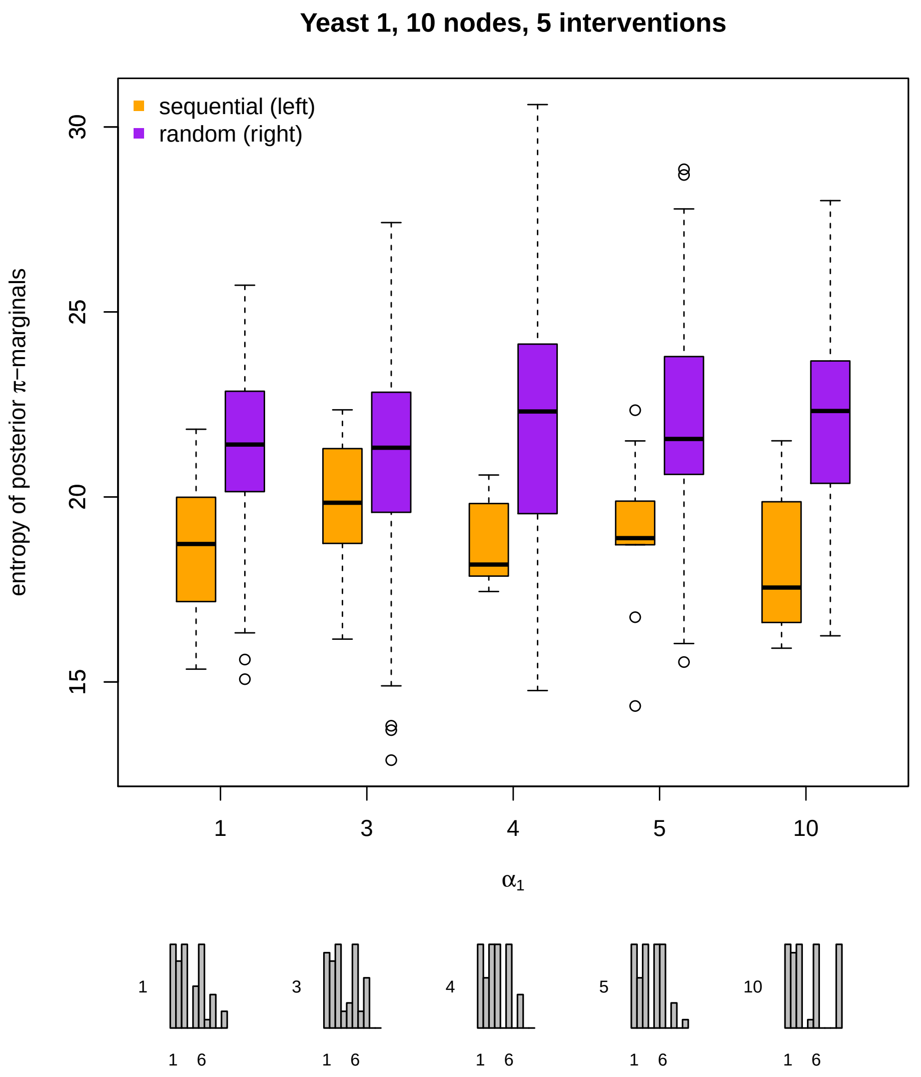


Figure 2.14: *Active learning performance at step 5 on Yeast 1. Top:* Box plots comparing the entropy of the posterior π-marginals after five steps on the 10-node Yeast 1 network. Each pair of box plots compare the active learning approach described in section 2.4 (right) to randomly chosen interventions (left). The label on the horizontal axis indicates the node targeted by the first intervention. *Bottom*: For each initial action, α_1, the bar plots show how many times each possible intervention occurred among the first five across 10 simulated data sets.

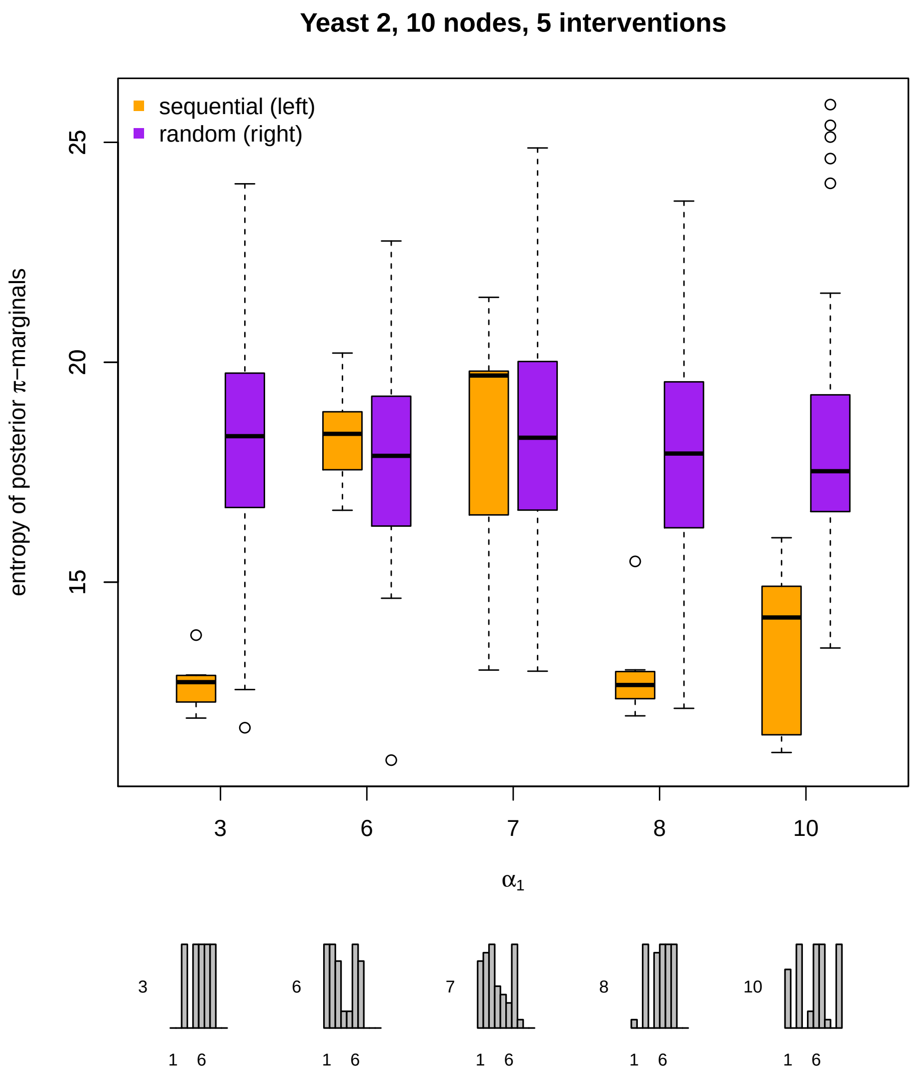


Figure 2.15: *Active learning performance at step 5 on Yeast 2. Top:* Box plots comparing the entropy of the posterior π-marginals after five steps on the 10-node Yeast 2 network. Each pair of box plots compare the active learning approach described in section 2.4 (right) to randomly chosen interventions (left). The label on the horizontal axis indicates the node targeted by the first intervention. *Bottom*: For each initial action, α_1, the bar plots show how many times each possible intervention occurred among the first five across 10 simulated data sets.

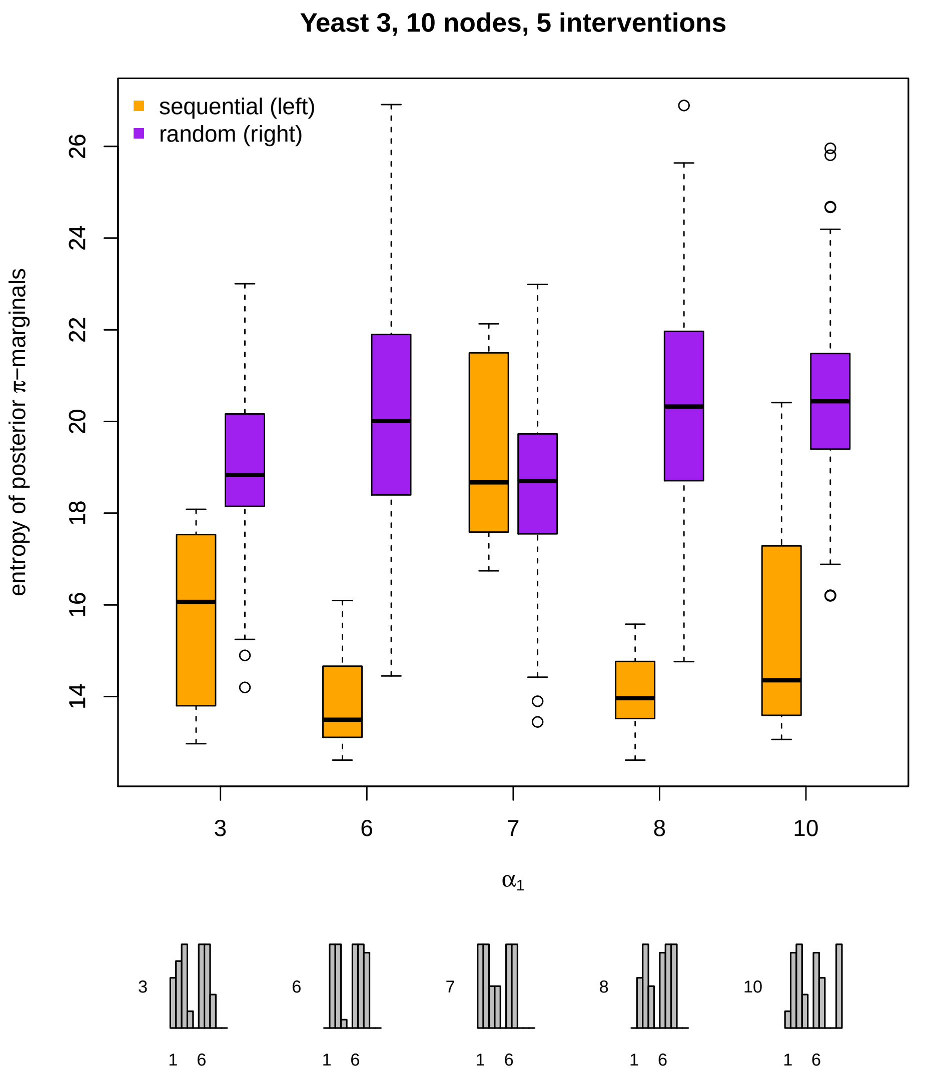


Figure 2.16: *Active learning performance at step 5 on Yeast 3. Top:* Box plots comparing the entropy of the posterior π-marginals after five steps on the 10-node Yeast 3 network. Each pair of box plots compare the active learning approach described in section 2.4 (right) to randomly chosen interventions (left). The label on the horizontal axis indicates the node targeted by the first intervention. *Bottom*: For each initial action, α_1, the bar plots show how many times each possible intervention occurred among the first five across 10 simulated data sets.

2.6 Performance on Benchmark Data

The DREAM competitions provide useful benchmark data for assessing the performance of network reconstruction methodologies [53, 52, 70]. In this section, we use data from DREAM 3, challenge 4, to assess the network reconstruction performance of the path-length model (2.3) when fit using the MCMC techniques of section 2.3.3. There are five 10-node networks in this benchmark data with realistic topologies; two extracted from the gene regulatory network of *E. coli*, labeled Ecoli 1 and Ecoli 2, and three extracted from the gene regulatory network of *S. cerevisiae*, labeled Yeast 1-3. The benchmark data is *in silico* but is simulated from a set of stochastic differential equations based on thermodynamics modeling both gene transcription and protein translation [53].

Datasets for each network in the DREAM 3 competition consist of a single observation per gene for wild-type, as well as all knockouts and all knockdowns. Using the computational model discussed above, knockouts are simulated by setting the expression level of the target gene to zero while knockdowns are simulated by halving the target's transcription rate. The datasets also contain time-course data not used in this chapter.

The difference between the knockout and knockdown data is accounted for by modifying the prior on the relative magnitude of effects given in (2.3). Recall that the latent β_i^α determines the relative position of μ_i^α between μ_i^0 and either zero or one, depending on ψ_i^α. For knockouts, the prior used is $\beta_i^\alpha \sim \text{Beta}(2, 2 + \gamma_i^\alpha)$. The prior for knockdowns is set so that at $\gamma_i^\alpha = 1$, i.e. i a child of α, the expected value of β is half what it is for a knockdown. This gives a prior of $\beta_i^\alpha \sim \text{Beta}(2, 6 + 2\gamma_i^\alpha)$.

The path-length model given in (2.3) assumes the noise variance σ^2 and the wild-type means μ_i^α are known. Since these are unavailable in the competition datasets the following plug-in estimates are used. First, the wild-type means for each gene are estimated by:

1. concatenating observations of that gene across all experiments in which it was not the intervention target;

2. discarding those observations more than 1.5 times the inter-quartile range from the median;

3. using the median of the remaining observations as the estimate.

Similarly, the standard deviation σ is estimated by:

1. concatenating the differences between the observed values and the observed wild-types across all experiments in which a particular gene was not the target;

2. discarding those observations more than 1.5 times the inter-quartile range from 0;

3. taking one-half the difference between the sample quantiles at $\Phi(1)$ and $\Phi(-1)$ as the estimate. Here Φ is the standard Normal cumulative distribution function.

Using these plug-in estimates, we draw $2e5$ samples from the posterior given all experiments of the path-length model using MCMC with partial-order proposals as described in section 2.3.3 with $\rho = .9$ and $p = .5$. The first $1e5$ samples are thrown out and the remainder used to estimate the posterior π- and γ-marginals. As in the competition, DAG reconstruction performance is evaluated using Area Under the Reciever Operator Characteristic (AUC-ROC) and Area Under the Precision Recall Curve (AUC-PR) with edges ranked using the appropriate posterior γ-marginal. Considering the focus in this chapter on partial orders we also compute AUC-ROC and AUC-PR for precedence relations with relations ranked according to the posterior π-marginals. Results for the 10-node networks are in Tables 2.3 and 2.4 and are presented alongside the top two performers from the original competition. We present results for the path-length model under two separate priors; one is uniform over DAGs the other over partial orders. Given the number of experiments available there are only small performance differences between the two, though the partial-order prior does perform marginally better. Based on these results and earlier simulation studies, we suggest the partial-order prior be used as a default.

	Ecoli 1	Ecoli 2	Yeast 1	Yeast 2	Yeast 3
Partial-Order Prior	0.989	0.998	0.974	0.835	0.648
DAG Prior	0.929	0.982	0.975	0.791	0.645
Partial-Order Prior	**0.827**	**0.916**	0.831	**0.677**	0.542
DAG Prior	0.699	0.868	0.812	0.649	0.534
Team 315	0.710	0.713	**0.897**	0.541	**0.627**
Team 291	0.544	0.748	0.771	0.352	0.493

Table 2.3: *AUC-PR comparisons on the DREAM 3 10-node networks.* The top two rows are the performance of our model in reconstructing the partial order. The bottom four rows are for reconstructing the DAG; the best performer among these values is in bold.

	Ecoli 1	Ecoli 2	Yeast 1	Yeast 2	Yeast 3
Partial-Order Prior	0.997	0.999	0.983	0.830	0.727
DAG Prior	0.981	0.992	0.984	0.784	0.711
Partial-Order Prior	**0.963**	**0.979**	0.941	**0.788**	0.672
DAG Prior	0.922	0.962	0.938	0.758	0.662
Team 315	0.928	0.912	**0.949**	0.747	0.714
Team 291	0.794	0.856	0.944	0.590	**0.715**

Table 2.4: *AUC-ROC comparisons on the DREAM 3 10-node networks.* The top two rows are the performance of our model in reconstructing the partial order. The bottom four rows are for reconstructing the DAG; the best performer among these values is in bold.

2.7 Conclusion

This chapter makes a number of contributions to the network reconstruction literature. First, we introduce a new class of models for intervention data based on path-lengths as an alternative to conditional independence models. Though the set of all minimum path lengths uniquely determine the DAG, path-length models shift focus from the existence of individual edges to the existence of precedence relationships. Whereas conditional independence models necessarily focus on parent-child relationships, path-length models emphasize ancestor-descendent relations. While such models lack the broad scope of conditional independence models and do not offer the same seamless link between observational and intervention data, they have the advantage of being well-suited to settings where available data is incomplete for a full network reconstruction.

This shift in focus from edges to paths is further developed by the introduction of novel prior distributions over the space of DAGs based on partial orders. While still non-informative, specifying a prior uniform over partial orders instead of DAGs results in more natural prior marginals which place greater and non-vanishing weight on a given pair of nodes having no order relation. In benchmark datasets the new prior resulted in better network reconstruction as measured by AUC-PR and AUC-ROC. Another benefit to the new prior is that it allows for a different take on sparsity; that is, sparsity in terms of feedforward network motifs rather than strictly in terms of edges.

Continuing the theme, an active learning framework based on greedily minimizing the average marginal entropy of edges was adapted to this chapter's focus on partial orders by instead greedily minimizing the marginal entropy of precedence relations. While no direct comparisons between these two approaches are carried out the new approach is better aligned to this chapter's goals and focus. In addition, a flexible and highly parallelizable Monte-Carlo scheme for estimating the proposed improvement function is developed.

Moreover, active learning aside, in settings with incomplete data for which individual edges are not well identified it can be argued that these posterior π-marginals provide a better summary than the corresponding edge posteriors. By moving up one level abstraction, the posterior π-marginals make it easier to see the structural information contained in incomplete data. At the very least, such a structural summary provides a useful complement to finer grained edge summaries.

In order to estimate these posterior π-marginals or any other posterior summary it is necessary that we be able to efficiently sample the posterior distribution. The complexity and geometry of the space of DAGs make this a challenging problem and this challenge is often a limiting factor in the applicability of Bayesian approaches to network estimation. In settings with incomplete data the challenge is that much greater as the posterior generally has multiple modes that, although structurally similar under path-length models, are not 'close' in the geometry resulting from the standard approach to sampling the posterior. This challenge is alleviated to some extent by the introduction of a novel proposal distribution for use within the Metropolis-Hastings algorithm. The new proposal is based on sampling DAGs according to the partition induced by partial orders effectively altering the induced geometry to better match the posteriors being sampled. The new proposal is shown to improve mixing compared to the standard approach and allows for Bayesian estimation on somewhat larger networks than is typically feasible.

CHAPTER III

Penalized Likelihood Estimation for Directed Acyclic Graphs with Incomplete Partial-Order Information

3.1 Introduction

There is a large literature on using penalized likelihoods to estimate the structure of graphs. This is especially true for undirected Gaussian graphical models in which the nodal variables are assumed to follow a multivariate normal distribution. In this setting, nodes not connected by an edge correspond to variables that are conditionally independent given the others. In the multivariate normal distribution two variables are conditionally independent if and only if the corresponding entry of the inverse covariance, or precision, matrix is zero. Edges thus correspond to nonzero entries in the inverse covariance matrix. Covariance estimation dates to Dempster [17] and has grown into a well-developed field. In high-dimensional settings a regularized estimate is often necessary. This is generally accomplished by assuming the precision matrix to be sparse; that is to have few non-zero entries relative to the total number of entries and the number of available samples. Some notable contributions to estimation of sparse inverse covariance matrices using ℓ_1-regularized likelihood include the graphical lasso [22], neighborhood selection [56], the consistency results in [75] and [46], and also

[48, 94, 3, 42]. Of particular relevance are methods penalizing entries in the Cholesky factor assuming a natural ordering of the variables [48, 42] as this chapter deals with the incorporation of order information into the estimation of directed networks. Penalties based on extensions of the BIC have also been studied [21, 25].

There is an important distinction between the approach taken in many of the papers just cited and the neighborhood selection approach [56]. The former jointly estimate the entries of the precision matrix while neighborhood selection estimates the dependencies node by node using the lasso [84]. The latter is much simpler computationally and remains consistent owing to the consistency of the individual regressions. However, there is a loss in efficiency as neighborhood selection does not make use of global properties such as the positive definiteness of the target covariance matrix. This is a theme we will return to when considering different ways to incorporate order information into estimates for directed graphs.

Turning to directed graphs, there is an equally substantial literature on estimating their structure. One prominent approach based on conditional independence tests is the PC-algorithm [82] shown to be consistent in an appropriate sense for high-dimensional ($p > n$) multivariate Gaussian distributions in [44]. To be precise, they show that the PC-algorithm consistently estimates the undirected skeleton provided the maximum degree in the true graph is smaller than the sample size n. Other approaches based on searching the space of DAGs to minimize a penalized likelihood score such as BIC have also been shown to be consistent; for the low-dimensional case with p fixed and $n \to \infty$ in [12] and for the high-dimensional case by [86]. As discussed in the previous chapter, there is also a wealth of literature on estimating directed graphs, i.e. Bayesian networks, when the nodes of the network represent discrete random variables.

These consistency results are not for the DAG itself, but for the Markov equivalence class consisting of DAGs encoding the same set of conditional independence relations, hence the same statistical model [87]. Graphs in the same Markov equivalence class have the same undirected skeleton and the same set of v-structures, $i \rightarrow k \leftarrow j$. An equivalence class of this sort can be characterized by its essential graph, also called a Completed Partially Directed Acyclic Graph (CPDAG), in which undirected edges $i - j$ indicate nodes for which $i \rightarrow j$ in some members of the equivalence class and $i \leftarrow j$ in others [2, 37]. This concept has been extended to interventional equivalence [37].

Even when excluding cyclic graphs, the space of DAGs has super-exponential complexity. On account of this, much of the literature is devoted to search algorithms for finding high scoring graphs. Of particular relevance are Greedy Equivalence Search [12] and Greedy Interventional Equivalence Search [37] both of which search over equivalence classes instead of DAGs. In general, node-wise approaches akin to neighborhood selection are inappropriate as the estimated graph may not be acyclic even after transforming bi-directed to undirected edges.

However, when the nodes are accompanied by a linear ordering both cycles and observational equivalence are avoided allowing the network to be built up from estimates of the parents of each node. This approach is studied in [79] in which the authors characterize the global optimization problem as a series of node-wise ℓ_1-penalized regressions and prove consistency by adapting the ideas of [56] to the context of directed networks. In simulations they show this method outperforms the PC-algorithm and a naïve application of the graphical lasso. This raises the question of how other types of order information can be brought to bear on estimation of directed networks.

To study this question, we first define so-called incomplete partial orders as a tool for incorporating order information into penalized-likelihood estimates of directed graphs. The main idea is to provide a framework for restricting the space of allowable DAGs by joining restrictions on the space of potential node orderings with information on the existence of (unknown) paths connecting pairs of nodes. Both types of restriction arise from interventions of the type studied in the previous chapter and partial orders provide a natural description of such information. However, knowledge of the partial order structure may be incomplete, whence the designation *incomplete* partial orders. The information content of an incomplete partial order lies along a three-dimensional lattice and we present plots for visually comparing the available information of different incomplete partial orders. The special case of a known linear order with no additional structural information comprises a corner in this lattice.

After defining incomplete partial orders, two methods for incorporating this structural knowledge into BIC-penalized regression are considered. In the first, edges are jointly estimated using the structural information of incomplete partial orders to restrict the search over the space of DAGs. The second method discards some of the information content in order to simplify estimation using a neighborhood selection approach. Comparisons are made between and within methods at different points in the information lattice.

3.2 Incomplete Partial Orders

Let $G = (V, E)$ be a DAG and Π the partial order induced by reachability. As in previous chapters the edge set E is unknown. However we will assume incomplete information on the relations in Π stemming from a previous set of intervention experiments or other prior knowledge. We call this *incomplete information* rather than *partial information* to avoid confusion with the concept of a *partial ordering* of the nodes in V. To understand the difference, notice that for the complete partial order induced by reachability, Π, we have $E \subset \Pi$ so that the graph structure is known up to the presence or absence of feedforward motifs where $i \to j \to k$ and $i \to k$. When Π is known, a number of potential edges can be excluded as $(i, j) \notin \Pi$ implies $(i, j) \notin E$. Contrast this with the situation in which only a subset $P \subset \Pi$ of the

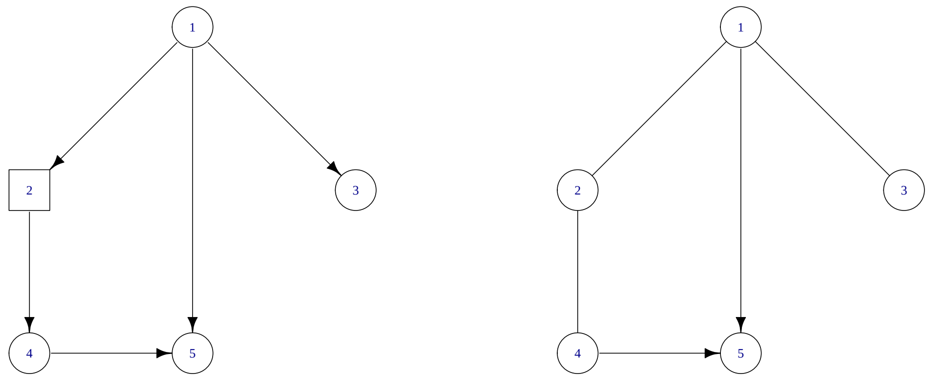


Figure 3.1: *Examples for the sets P and N defining an incomplete partial order.* In the DAG shown here on the left an intervention at node '2' may reveal that nodes '4' and '5' are descendants. This information is formally specified by setting $P = \{(2,4),(2,5)\}$. The intervention may also further restrict the space of DAGs by revealing that certain paths do not exist. Some possibilities for such negative constraints are: $N = \{(4,2),(5,2)\}$, $N' = N \cup \{(4,1),(4,3)\}$, and $N'' = N' \cup \{(2,1),(2,3),(4,1),(4,3),(5,1),(5,3)\}$. The pairs (P,N), (P,N') and (P,N'') are all valid incomplete partial orders consistent with the DAG on the left. On the right is the essential graph or CPDAG representing the equivalence class to which the DAG on the left belongs.

precedence relationships are known so that $(i,j) \notin P$ does not imply $(i,j) \notin E$. The following notations will be used to define *incomplete partial orders*. First, there is a set of known (positive) relations $P \subset \Pi$. There is also a set of relations, $N \subset \Pi^c$, known to be null and consisting of ordered pairs of nodes known not to be connected by any directed path. Several examples of an incomplete partial order consistent with a specific DAG are given in FIGURE 3.1. P is required to be a partial order and its transitive reduction is denoted by $\check{P}$. Moreover, since G and Π are acyclic, we will require that relations (i,j) which would make P cyclic be included in N. For instance, for any P we have $(a,b) \in P$ implies $(b,a) \in N$ and also $(a,a) \in N$ for all $a \in V$. Likewise, we require a sort of transitivity so that if $(a,b) \in P$ and $(a,c) \in N$ then also $(b,c) \in N$ so that there are no implicit restrictions on the associated partial orders. These ideas are collected in the following definition.

Definition 3.1. Let Π be a (strict) partial ordering of a set V. Then the ordered double (P, N) is called an *incomplete partial order with respect to* Π if the following conditions are satisfied:

1. $P \subset \Pi$ and is itself a (strict) partial order; i.e. P is irreflexive, transitive, and antisymmetric.

2. $N \subset \Pi^c$ and $(i, j) \in N$ for any $(i, j) \in V \times V$ such that $P \cup \{(i, j)\}$ is not a partial order.

3. If $(h, i) \in P$ and $(h, j) \in N$ then $(i, j) \in N$.

For the complete partial order Π induced by the precedence relationships in $G = (V, E)$ we know the transitive reduction $\check{\Pi}$ is a subset of the edge set E since these relations are not mediated by other edges in E. Though the same cannot be said for an incomplete partial order (P, N), there is an analogous statement. The key is that there must be an edge from i to j for $(i, j) \in \check{P}$ if there are no potential mediators among the unknown relations. Formally, for (P, N) an incomplete partial order with respect to a Π, $(i, j) \in \check{P}$ implies $(i, j) \in \check{\Pi}$ if and only if for all $k \in V$, $(i, k) \in N$ or $(k, j) \in N$. Call the collection of all such relations *mandatory* and denote the set of mandatory relations by $M = M(P, N)$. Call (P, N) *strong* if $M = \check{P}$ and *weak* otherwise.

A final bit of notation will be useful. For $a \in V$ and an incomplete partial order (P, N), let $P_a = \{b : (b, a) \in P\}$ denote the known ancestors of a, $N_a = \{b : (b, a) \in N\}$ denote nodes which must not precede a, and $M_a = \{b : (b, a) \in M\}$ the mandatory edges known to be in E.

3.3 Comparing Types of Order Information

As discussed in the introduction, previous work has considered penalized likelihood estimation of DAGs given a known linear ordering of the nodes [78]. Observational

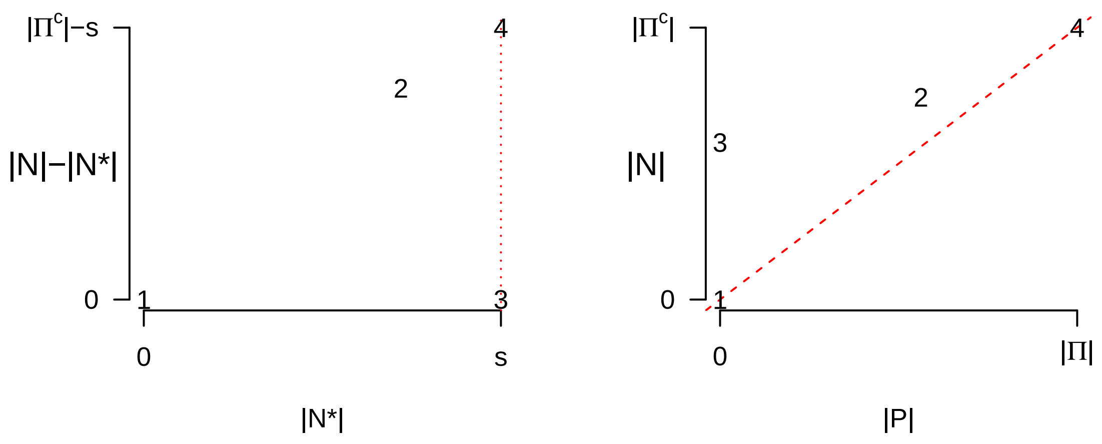


Figure 3.2: *Visualizing the lattice of order information.* The information content of an incomplete partial order (P, N) with respect to a complete partial order Π can be summarized using $|P|$, $|N|$, and $|N^*|$. On the right four incomplete partial orders labeled 1-4 are plotted in $|P| - |N|$ space showing the relative size of the known ancestral relations P and the forbidden ancestral relations N. The red line is there as a reminder that $|N| \geq |P|$ in all cases. On the left, the magnitude for the set of forbidden ancestral relations is split into $|N^*|$, the number of pairs with at least one forbidden direction, and the remainder $|N| - |N^*|$ indicating the number of pairs not connected by a directed path in either direction. Incomplete partial orders on the vertical line at $s = \binom{d}{2}$ correspond to DAGs completely determined by their skeletons. The plotted incomplete partial orders are: 1. no information, $(\emptyset, \emptyset)$; 2. incomplete information; 3. a linear ordering, $P = \emptyset$, $N = L^c$; and 4. the complete partial order, (Π, Π^c).

equivalence is no longer an issue in this setting as the direction of any edge is fixed *a priori* so that the estimation task is significantly simplified. A known linear order of the nodes is a special case of an incomplete partial order. Namely, if L is a linear order then the corresponding incomplete partial order is $(P, N) = (\emptyset, L^c)$ where the complement is with respect to $V \times V \setminus \{(a, b) : a = b\}$. Notice that this places no restriction on the undirected skeleton.

By expanding either P or N it is possible for an incomplete partial order to carry more information than a linear order of this type by further restricting the space of allowable DAGs. For instance, suppose two linear orders L_1 and L_2 are known to be linear extensions of Π and take $N = L_1^c \cup L_2^c$. For a concrete example, consider again the DAG in 3.1 and let $\sigma_1(V) = 12345$ be the identity permutation, $\sigma^{-1}(i)$ indicate the location of node i in the permutation, and define $L_1 = \{(i, j) : \sigma_1^{-1}(i) < \sigma_1^{-1}(j)\}$. Likewise, if $\sigma_2 = 13245$ then $L_2 = \{(i, j) : \sigma_2^{-1}(i) < \sigma_2^{-1}(j)\}$ is another linear extension. In this case, we have $L_1 \cup L_2 = L_1 \cup \{(3, 2)\} = L_2 \cup \{(2, 3)\}$ and any incomplete partial order $(\cdot, L_1^c \cup L_2^c)$ disallows DAGs with a precedence relationship between nodes '2' and '3'. Moving in the opposite direction, an incomplete partial order can also carry less information than a linear order with an extreme example being $(P, N) = (\emptyset, \emptyset)$ where no order information is available. Useful summaries of the information regime are given by the cardinalities of P and N. At the population level these quantities are bounded by, $0 \leq |P| \leq |\Pi|$ and $0 \leq |N| \leq |\Pi^c| = d(d-1) - |\Pi|$. In practice, $|\Pi|$ is unknown but the above bounds are useful for understanding simulation results.

Much of the appeal of working from a known linear order comes from the fact that the direction of all edges is identifiable for any skeleton. Since L is a linear order, for all $a, b \in V$ either $(a, b) \in L$ or $(b, a) \in L$ and the same is true of the complement $N = L^c$. For instance, consider the CPDAG on the right of Figure 3.1 which has three edges whose directions are unidentifiable solely from conditional independence information. However, if an incomplete partial order (P, N) is such that $(3, 1) \in N$ then the direction of the edge connecting nodes '1' and '3' becomes identified. When L is a linear extension of the DAG and $L^c \subset N$, clearly the direction of all edges is identified.

Defining the set of unordered pairs $N^* = \{\{a, b\} : (a, b) \in N \text{ or } (b, a) \in N\}$, for any incomplete partial order (P, N) the condition $|N^*| = \binom{d}{2}$ is clearly sufficient for this sort of *a priori* identifiability as all possible edges are then limited to having a single direction. Less clear is whether this condition is necessary; i.e. do there exist (P, N) with $|N^*| < \binom{d}{2}$ for which two distinct but observationally equivalent DAGs are consistent with the order information? A partial answer is available when $P = \emptyset$; in which case $|N^*| = \binom{d}{2}$ is necessary for identifiability of any skeleton from observational data alone.

Together the magnitudes $|P|$, $|N|$, and $|N^*|$ summarize the different sorts of information carried by an incomplete partial order (P, N). Since each of these quantities has integer values the information content of (P, N) lies at some point in a finite three-dimensional lattice. The locations of different partial orders in this lattice can be compared using a pair of two-dimensional plots as in FIGURE 3.2. The figure on the right shows the number of ordered pairs in P and N respectively. Since we always have $|N| \geq |P|$ the region below the diagonal is empty. Similarly, in the figure on the left relating the magnitudes of N and N^* the vertical axis is rescaled to omit the empty region where $|N| < |N^*|$. The vertical line at the rightmost extreme of this plot indicates the plane in the lattice along which at least one linear ordering of the nodes is known. See section 3.5 for a detailed description of the incomplete partial orders 1-4 indicated in the figure.

3.4 Penalized Likelihood Estimation

Consider data arising from a linear structural equation model consistent with a DAG G,

$$Y_i = \sum_{j \in \mathrm{pa}_i(G)} \beta_{ij} Y_j + \epsilon_i \tag{3.1}$$

with $\{\epsilon_i\}$ independent, $\mathbb{E}[\epsilon_i] = 0$, $\mathrm{var}(\epsilon_i) = \sigma^2$ a known constant, and $\mathrm{pa}_i(G)$ the parents of i in G. Form the vector $\boldsymbol{Y} = (Y_1, \dots Y_d)'$, let $\mathcal{D} = (\boldsymbol{Y}_k)_{k=1}^n$ collect multiple realizations over the noise, and denote the log-likelihood by $\ell(\beta; \mathcal{D})$. In this setting, the log-likelihood can be split into a summation over terms for each node,

$$\ell(\beta; \mathcal{D}) = \sum_{i=1}^{d} \ell(\beta_i; \mathcal{D}). \tag{3.2}$$

This is also the case with unknown variance provided $\mathrm{var}(\epsilon_i) = \sigma_i^2$, i.e. each noise variables has its own variance. In this setting we should replace β_i with $\theta = (\beta_i, \sigma_i)$ and β with θ in the log-likelihood.

In order to respect the information given by the incomplete partial-order (P, N) we need to restrict the allowed sparsity patterns of $\beta = [\beta_{ij}]$. For any (complete) partial order Π, let

$$S(\Pi) = \{\beta \in \mathbb{R}^{d \times d} : \beta_{ij} \neq 0 \iff (j, i) \in \Pi\}, \tag{3.3a}$$

and

$$S_i(\Pi) = \{\beta \in \mathbb{R}^{d} : \beta_{j} \neq 0 \iff (j, i) \in \Pi\}. \tag{3.3b}$$

The sets $S(\Pi)$ and $S_i(\Pi)$ represent all real matrices/vectors whose nonzero entries are restricted by the ancestral relations in Π. Other useful sets include the set of all (complete) partial orders consistent with an incomplete partial order (P, N),

$$\Pi(P, N) = \{\Pi \subset V \times V \text{ a partial order} : P \subset \Pi, \Pi \subset N^c\}, \tag{3.4}$$

and the set of graphs with bounded in-degree,

$$\mathcal{G}_k = \{G = (V, E) \text{ a DAG} : \forall\, b \in V, |\{a \in V : (a, b) \in E\}| \leq k\}. \tag{3.5}$$

Write $\Pi(P, N, k) = \{\Pi \in \Pi(P, N) : \check{\Pi} \in \mathcal{G}_k\}$ for the set of all partial orders consistent with (P, N) whose transitive reductions have maximum in-degree k. Similarly $S_i(\Pi, k)$ and $S(\Pi, k)$ bound the maximum number of nonzero entries for each $\beta_{i\cdot}$.

3.4.1 Using ℓ_0-penalties for graphs with bounded in-degree

Having defined the sets above a standard approach to model selection using BIC [76] can be formulated as the following penalized-likelihood problem,

$$\hat{\beta} = \arg \min_{\pi \in \Pi(P,N,k)} \arg \min_{\beta \in S(\pi,k)} -2\ell(\beta; \mathcal{D}) + ||\beta||_0 \log n. \tag{3.6}$$

The problem above is equivalent to minimizing BIC over all DAGs in $\mathcal{G}_k$ consistent with (P, N), but the decomposition into separate inner and outer optimizations offers insight into how it will be carried out. Beginning with the inner optimization, observe that because the likelihood and penalty split into separate sums for each node, the optimization problem can be decomposed along the same lines. That is, if

$$\hat{\beta}_{i\cdot} = \arg \min_{\beta \in S_i(\pi,k)} -2\ell(\beta; \mathcal{D}) + ||\beta_{i\cdot}||_0 \log n \tag{3.7}$$

then $\hat{\beta} = [\hat{\beta}_{i\cdot}]$.

By limiting the number of nonzero entries in each $\beta_{i\cdot}$ to sufficiently small k, for moderate network dimension d, the individual regressions (3.7) are solvable by exhaustive search. Specifically, for each of $\sum_{j=1}^{k} \binom{d}{j}$ subsets, of crude order $O(d^k)$, the most expensive operation in solving the relevant regressions is inverting a $k \times k$ or smaller matrix. These can be precomputed and stored so that when performing the outer optimization the scores can be found by looking for minima over appropriate index sets.

While the inner optimization is solvable using standard methods, the outer problem is a difficult combinatorial optimization. We tackle it using a stochastic search modeled on the MCMC procedure in section 2.3.3. The basic idea is to run the Metropolis-Hastings algorithm outlined there with the likelihood replaced by -1 times the scores minimizing the inner optimization above. One difference is that only partial orders are sampled while DAGs are determined by solving the individual ℓ_0-penalized regressions as just discussed. Another difference is that we are using Metropolis-Hastings for stochastic optimization rather than for sampling a posterior so that while running the Markov chains the lowest scoring β's are retained including those among rejected proposals. The proposal distribution should also be adjusted so that proposals are confined to $\Pi(P, N, k)$ but for simplicty we assign an infinite score to graphs outside this region.

3.4.2 Neighborhood Selection

A neighborhood-selection approach solves each of the single-node regressions (3.7) over the space of vectors with less than k non-zero entries in $N_i^c \cup \{i\}$. This discards all of the information in P. In general, it can also fail to prevent paths in N from appearing in the estimate as this approach only uses N to restrict parent and not ancestral sets. However, there are instances for which this is not a concern. One class of such exceptions occurs when for all i, $j \in N_i$ implies $N_i \subset N_j$. An example from this class is given by the second incomplete partial order described in the next section.

Another important class of exceptions occurs when $|N^*| = \binom{d}{2}$ so that at least one linear ordering of the nodes is available and an estimated network is fully determined by its skeleton. In these settings the global problem is solved by neighborhood selection making it preferable to stochastic or greedy search. Finally, in situations where the Markov equivalence classes are non-trivial and contain more than one graph neighborhood selection can still be used to estimate the skeleton. The following section compares stochastic search and neighborhood selection under different incomplete partial orders.

3.5 Numeric Work

In this section we evaluate the impact of incomplete partial order information on network reconstruction performance. Specifically we will compare performance under four information regimes: 1. no order information a la [56], 2. an incomplete partial order (P, N) described below, 3. a linear order a la [78], specifically the canonical order, and 4. the complete partial order Π. Comparisons are made on the fixed 25-node DAG shown in Figure 3.3 using data simulated from the model (3.1) with $\epsilon_i \overset{iid}{\sim} N(0,1)$ and $\rho_{ij} \sim \text{Uniform}(.2, 1)$ following the setup in [44]. There are 50 simulation runs with both ρ and ϵ redrawn in each run. Estimates using stochastic search take the minimum BIC over a thousand steps in the Metropolis-Hastings algorithm and use $p = .5$ for sampling severed paths. All chains in the stochastic search had acceptance rates from 5-20%. For both the stochastic search and neighborhood selection approaches the search is restricted to graphs with a maximum in-degree of three.

The second incomplete partial order (2) can be viewed as arising from interventions to nodes 1 thru 4 at the population level, i.e. without sampling errors. The descendant sets for these interventions are $D_1 = \{2, \ldots, 24\}$, $D_2 = \{5, \ldots, 10\}$, $D_3 = \{23, 24, 25\}$,

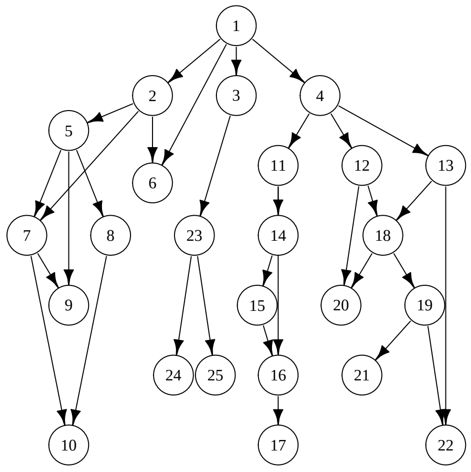


Figure 3.3: *25-node network used to compare information regimes.* Four information regimes are specified on this network using incomplete partial orders. Edges connect 32 of 300 pairs of nodes. The first incomplete partial order is empty giving no information. The second arises from interventions to nodes 1-4 and partitions the graph into descendants of nodes 2, 3, and 4. The edges $1 \to 2$, $1 \to 3$ and $1 \to 4$ are known *a priori* in this regime. The third gives a linear ordering with no additional information. The fourth gives the complete partial order under which all but the six feedforward edges are given.

and $D_4 = \{11, \ldots, 22\}$. The incomplete partial order (P, N) is given by,

$$P = \biguplus_{a=1}^{4} \{(i,j) : j \in D_a\}, \quad N = \biguplus_{a=2}^{4} \{(i,j) : i \in \bar{D}_a, j \in \bar{D}_a^c\} \tag{3.8}$$

where $\bar{D}_a^c = \{a\} \cup D_a$.

In TABLE 3.1 for σ known and TABLE 3.2 for σ estimated we compare the information regimes in terms of the True Discovery Rate (TDR) defined as the ratio of correctly estimated (directed) edges to the total number of estimated edges. Under incomplete partial orders 1 and 2 the directions of estimated edges are non-identified when using the neighborhood selection approach; bi-directed are treated independently so that edges in the true DAG but estimated as bi-directed are counted as one true positive and one false positive. With σ estimated the direction of these edges are also non-identified in the model. For these information regimes we also provide the TDR for the skeleton as separate rows in the table; the skeleton is estimated by including edges that appear in either or both directions. Incomplete partial orders two and four have nonempty P of which 2 and 26 ordered pairs, respectively, are 'mandatory' edges meaning they can be identified from (P, N) alone. Corrected TDRs excluding these edges appear to the right of the overall TDRs.

Reconstruction performance under the four information regimes is further compared in terms of precision and recall at the four different sample sizes in FIGURE 3.4 for σ known and FIGURE 3.5 for σ estimated. Precision is the same as the TDR while recall is the ratio of true positives to both true and false positives. All values are uncorrected for edges given *a priori* from the information regime and, as in the tables, bi-directed edges from neighborhood selection are treated independently.

Looking at the tables, we see that under all information regimes performing a search over partial orders is uniformly worse than the neighborhood selection approach. This is true for both the directed and undirected TDRs and holds whether σ

is known or estimated. The difference is most pronounced in the case of no information but is evident under regimes two and three as well and also increases somewhat with sample size. The box plots in FIGURES 3.4 and 3.5 tell a similar story with the exception that the skeletal recall under incomplete partial order two is marginally better under stochastic search.

For a known linear order as under information regime three, the difference in performance is to be expected as neighborhood selection solves the global optimization problem in a single sweep. It is, however, surprising with no or incomplete order information as under regimes two and three. In part this likely reflects a failure of the stochastic search to find a global minimum. As evidence consider that neighborhood selection improves more rapidly with increasing sample size than does stochastic search as can be seen in the box plots. The difference in performance can also be attributed in part to the fact that neighborhood selection is not restricted to estimating a DAG so that including an incorrectly oriented edge in the estimated edge set does not preclude also including the correctly oriented edge. When the orientation of edges is weakly or non-identified this provides a substantial advantage.

Turning to comparisons of the information regimes, for reconstructing the directed network there is a clear ordering from 1 to 4 among the regimes as expected. The same ordering applies in terms of recall when estimating the skeleton but is less pronounced. Interestingly, the information regime corresponding to the second incomplete partial order achieves greater skeletal precision than does the case of a known linear order. This reflects the fact that the second incomplete partial order places more restrictions on the space of allowed edges than does incomplete partial order four; i.e. $N(2) > N(4)$ despite the fact that $N^*(2) < N^*(4)$ as indicated in FIGURE 3.2.

	Overall				Corrected			
n	20	25	50	100	20	25	50	100
1-search	0.20	0.22	0.24	0.25				
2-search	0.44	0.47	0.46	0.48	0.39	0.42	0.41	0.44
3-search	0.46	0.49	0.59	0.67				
4	0.86	0.88	0.92	0.94	0.45	0.50	0.65	0.76
1-NS	0.30	0.34	0.40	0.44				
2-NS	0.47	0.50	0.53	0.55	0.42	0.45	0.49	0.52
3-NS	0.53	0.59	0.72	0.80				
1-search (skeleton)	0.36	0.40	0.46	0.49				
2-search (skeleton)	0.58	0.61	0.64	0.67	0.54	0.58	0.61	0.65
1-NS (skeleton)	0.47	0.52	0.66	0.76				
2-NS (skeleton)	0.68	0.71	0.80	0.85	0.64	0.68	0.78	0.83

Table 3.1: *Comparison of true discovery rates with known error variance.* Average true discovery rates over 50 samples for each of the four information regimes. On the top are values from stochastic search using all order information while on bottom are results from the neighborhood selection approach. In regimes 2 and 4, respectively, 3 and 26 edges can be determined from the order information alone. Values excluding these are given as the corrected true discovery rate on the right. The standard errors of most estimates are close to .01 while the standard errors of the corrected TDR under the complete partial order (4) are around .03.

	Overall				Corrected			
n	20	25	50	100	20	25	50	100
1-search	0.18	0.19	0.23	0.28				
2-search	0.37	0.40	0.43	0.50	0.32	0.35	0.39	0.46
3-search	0.42	0.45	0.56	0.65				
4	0.83	0.87	0.88	0.92	0.41	0.49	0.57	0.69
1-NS	0.23	0.27	0.36	0.41				
2-NS	0.40	0.43	0.50	0.53	0.36	0.39	0.47	0.50
3-NS	0.45	0.51	0.65	0.77				
1-search (skeleton)	0.35	0.37	0.45	0.51				
2-search (skeleton)	0.54	0.57	0.64	0.69	0.50	0.54	0.61	0.67
1-NS (skeleton)	0.35	0.41	0.57	0.69				
2-NS (skeleton)	0.56	0.62	0.74	0.80	0.52	0.58	0.71	0.78

Table 3.2: *Comparison of true discovery rates with estimated error variance.* Average true discovery rates over 50 samples for each of the four information regimes. On the top are values from stochastic search using all order information while on bottom are results from the neighborhood selection approach. In regimes 2 and 4, respectively, 3 and 26 edges can be determined from the order information alone. Values excluding these are given as the corrected true discovery rate on the right. The standard errors of most estimates are close to .01 while the standard errors of the corrected TDR under the complete partial order (4) are around .02.

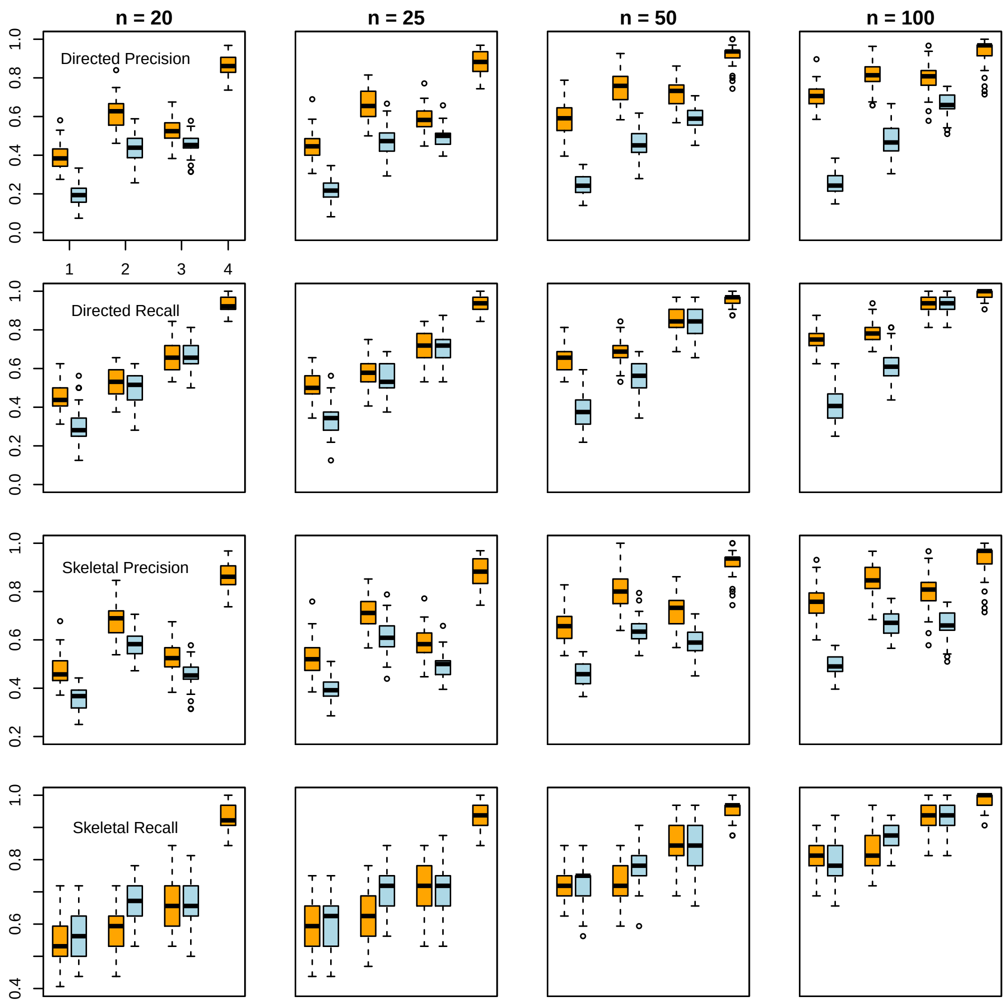


Figure 3.4: *Precision and recall comparisons of the information regimes with known error variance.* Each panel in the figure compares the reconstruction performance of the stochastic search and neighborhood selection approaches under the four information regimes. Each pair of box plots corresponds to an information regime with neighborhood selection on the left in orange and stochastic search on the right in blue. Each column corresponds to a sample size, $n \in \{20, 25, 50, 100\}$. Each row corresponds to a performance measure; from top to bottom: directed precision, directed recall, skeletal precision, and skeletal recall.

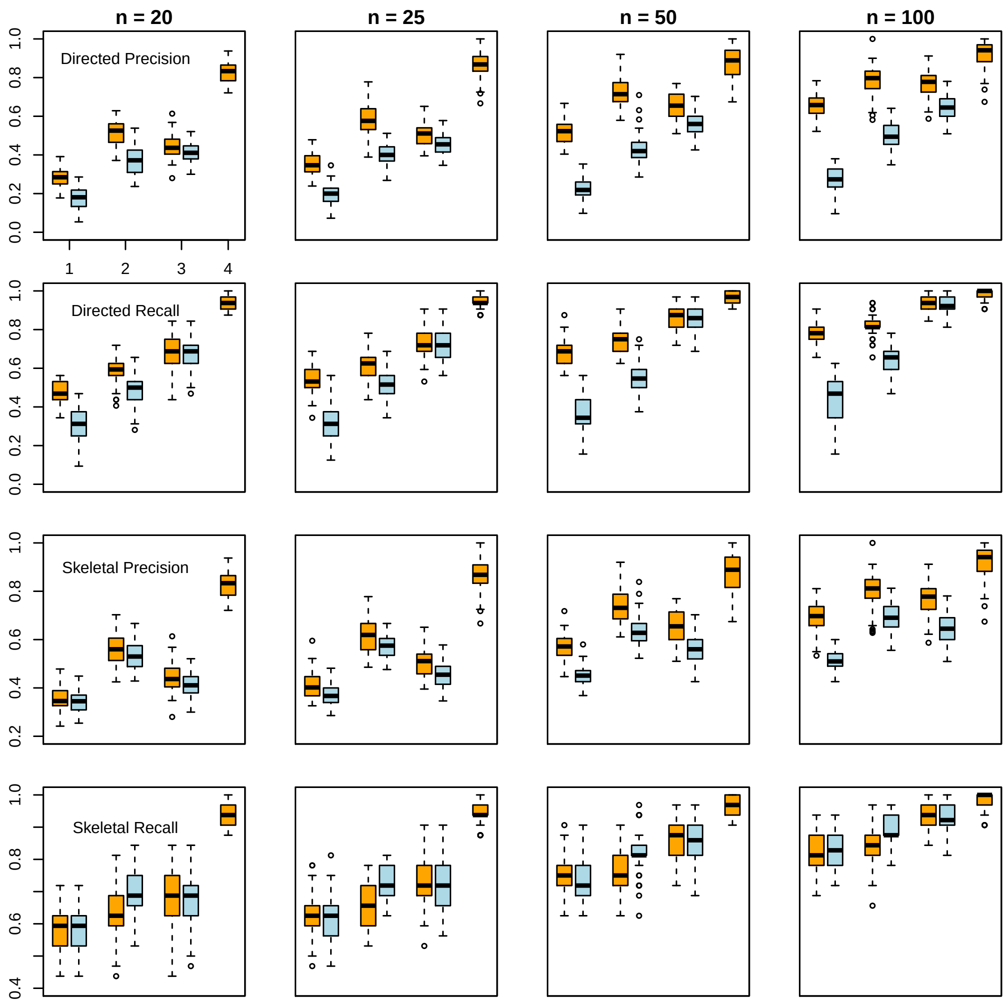


Figure 3.5: *Precision and recall comparisons of the information regimes with estimated error variance.*

3.6 Discussion

In this chapter we defined *incomplete partial orders* as a means for incorporating prior information on the ordering of variables when estimating the edge sets of DAGs. Incomplete partial orders are formulated in terms of known ancestral relations P and null relations N that restrict the space of DAGs over which penalized-likelihood estimation is performed. The information content of an incomplete partial order (P, N) lies along a three-dimensional lattice based on the magnitudes $|P|$, $|N|$, and $|N^*|$ where N^* is N viewed as a set of unordered pairs. A simple visualization based on a pair of two-dimensional plots allows the information content of different incomplete partial orders to be compared.

Ignoring the information in P, we investigate a neighborhood selection approach previously studied for estimating undirected graphical models and directed models given a linear ordering of the variables though using the ℓ_0-based BIC in place of an ℓ_1 lasso penalty. The reconstruction performance of this approach is compared under a variety of information regimes encompassing known linear orders as well as both weaker and stronger restrictions on the space of DAGs. Estimates using this approach show improved reconstruction performance under additional order information.

A second approach incorporating all of the information in (P, N) but requiring stochastic search over a restricted space of DAGs is also considered. While this approach has the advantage of using all available information and always returning a DAG the stochastic search algorithm is expensive and can lead to instability in the estimates. In a simulation study, this manifested as a failure to improve reconstruction performance relative to the neighborhood selection approach. Nevertheless this approach has the desirable property that the estimate is itself a DAG and also demonstrates improved performance with additional order information. Two directions for improvement come to mind. First, incorporating the order information into a greedy search over equivalence classes is likely to improve algorithmic and consequently estimation performance. A related direction for future research would be characterizing how order information affects identifiability as the authors in [37] do when stochastic interventions are directly available. Conceptually this is a simple matter of examining which DAGs lie in the intersection of an equivalence classes and an incomplete partial order (P, N). However, in practical terms it remains unclear how to characterize this intersection.

CHAPTER IV

Network Reconstruction Using Nonparametric Additive ODE Models

4.1 Introduction

4.1.1 Chapter Summary

Despite their utility, network reconstruction methods relying on conditional independence models or direct cause-effect methodologies have trouble with feedforward motifs and are are also generally limited to acyclic networks. For this reason, there is a need for approaches tailored to time-series data to augment those that rely on direct intervention experiments, especially as the former are often more readily available in some domains. In this chapter, we introduce an approach to reconstructing directed networks based on dynamic systems models. Our approach generalizes commonly used ODE models based on linear or nonlinear dynamics by extending the functional class for the functions involved from parametric to nonparametric models. Concomitantly we limit the complexity by imposing an additive structure on the estimated slope functions. Thus the submodel associated with each node is a sum of univariate functions. These univariate component functions form the basis for a novel coupling metric that we define in order to quantify the strength of proposed relationships and rank potential edges. We show the utility of the method by reconstructing networks using simulated data from computational models for the glycolytic pathway of *Lactocaccus Lactis* and a gene network regulating the pluripotency of mouse embryonic stem cells. For purposes of comparison, we also assess reconstruction performance using gene networks from the DREAM challenges. We compare our method to those that similarly rely on dynamic systems models and use the results to attempt to disentangle the distinct roles of linearity, sparsity, and derivative estimation.

4.1.2 Network Reconstruction using Time Course Data

Network reconstruction methods utilizing time-course data have the potential to avoid some of the limitations of both conditional independence models and direct cause-effect methodologies. For the former, dynamic versions of conditional independence models can accommodate feedback loops by allowing cycles to unfold over time. For instance, reconstruction methods based on statistical time series models such as (sparse) vector autoregression [77, 24] or state space models [92] fit into this framework, although they are not usually viewed this way [80]. Time-course data can also be used to orient edges after estimating an undirected network from perturbation experiments [78]. Moreover, time-course experiments under global perturbations, including environmental stressors such as heat shock, as well as changes in initial concentrations, are generally easier to carry out than knockouts and, though requiring a greater number of measurements, have lower setup costs. Finally, time-course methods are potentially useful not only for network reconstruction, but also for predicting the response of the system to yet to be observed perturbations.

Network formalisms for time-course data are closely related to regression-style methods of network reconstruction in that both treat the latter as a feature selection problem. In the generic case, regression-based formalisms seek models that express the observations associated with each node in terms of functions of observed values on other nodes. The edges of the network are determined by the variables these functions depend on. For time-course data, the regression model often take the form of a dynamic system expressed using ordinary differential equations (ODEs),

$$\dot{x}(t) = f(x(t)), \tag{4.1}$$

where the rate of change in system components $\dot{x}(t)$ is a function, f, of the component trajectories, $x(t)$. In this case, network reconstruction is a matter of finding the nonzero elements in the Jacobian $J = [\frac{\partial f_i}{\partial x_j}]$. Depending on the parametric form of f, finding the nonzero elements of J may reduce to finding nonzero parameters. A similar formulation is possible when (4.1) is replaced with a stochastic differential equation appropriate for single cell dynamics [64]. Though our focus is on time-course experiments, for completeness we note that nonlinear ODE models can sometimes also be fit by solving a related linear systems using data from perturbed steady states, potentially reducing the number of measurements required [57]. In the following subsection, we discuss ODE models for time-course data in greater detail.

4.1.3 Differential Equation Models

Due to their long history of successful application in modeling physical phenomena, ODEs provide an attractive class of models for time-course data. There are three main decisions to be made when developing and fitting an ODE model for network reconstruction from time-course data: the model class, an approach to parameter estimation, and finally a variable selection method for the actual network reconstruction. For instance the well-known Inferelator tool employs linear models, uses a (modified) gradient-matching approach, and in its original form employs ℓ_1 regularization for variable selection [5, 50, 29].

The first decision in developing an ODE model for time-course data is its parametric form. Typically the right-hand-side function (also called the slope function) is taken to be linear [49] or sigmoidal [93]. Linear ODE models are attractive because they allow one to use a number of specialized techniques, like the ability to combined multiple time-series from different experimenters [90]. While linear models also offer computational advantages and inferential simplicity, most biological processes are highly nonlinear. Nevertheless, as first-order approximations, linear models offer some protection against model misspecification. Alternatively, one can take a nonparametric approach in which the right-hand-side function is subject only to smoothness conditions. This flexible approach guards against model misspecification while allowing for nonlinearities. Other choices of models can be found in [13], though many of these have not been specifically applied to the network reconstruction task.

Approaches to estimating parameters in ODE models fall into two broad categories: trajectory matching and gradient matching. These two approaches differ in how they deal with the challenge of having equations describing the derivatives, but observations on the trajectories. The trajectory-matching approach involves choosing parameters minimizing some loss function, such as the sum of squared errors, measuring the discrepancy between a computed trajectory and the observations. If the trajectories—solutions to the initial value problem—are not available analytically, they can be found using numerical integration. The other approach, gradient-matching, instead first estimates the unobserved derivatives and then selects parameters minimizing a loss function measuring the discrepancy between the estimated derivatives and the right-hand-side function. An important feature of a gradient-matching procedure is how the derivatives are estimated. While trajectory matching is known to be statistically efficient [4] (the parameter estimates achieve a lower bound on the asymptotic variance) it can be computationally intractable for large networks. This often remains true even after one takes advantage of techniques such as differential elimination [61] for reducing the dimensionality of the system. Consequently, most ODE methods for network reconstruction employ a gradient-matching estimation scheme.

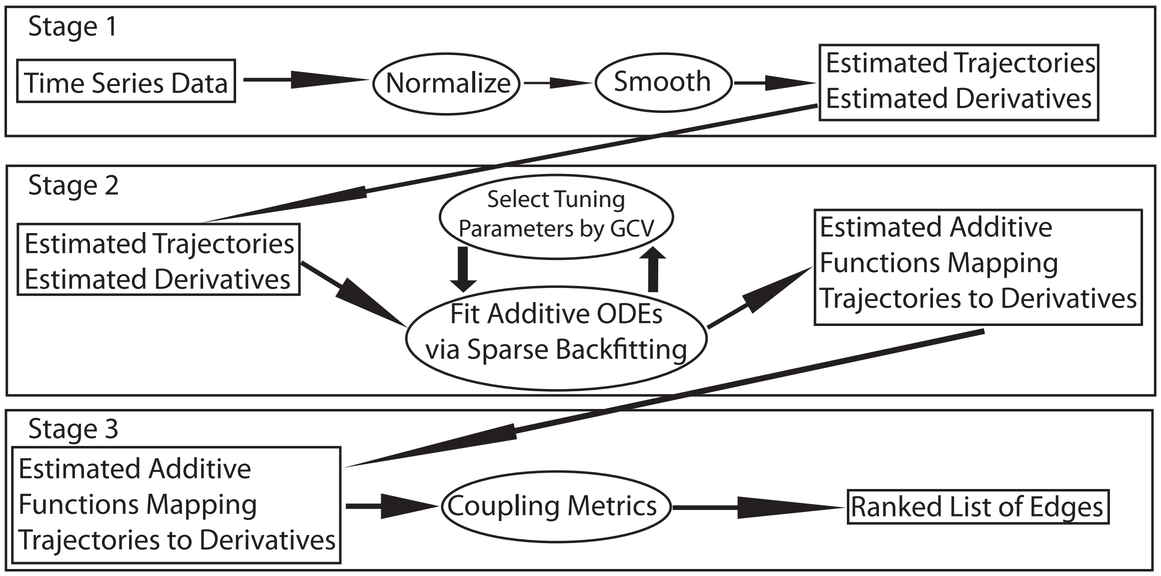
Fortunately, recent statistical work shows some gradient-matching procedures are also statistically efficient [73, 71] or nearly so [32].

The final and arguably most important decision in ODE-based network reconstruction is feature (variable) selection. After all, feature selection—deciding which components should appear in each of the right-hand-side functions—ultimately determines the estimated network. Feature selection often begins with a prescreening step in which the pool of potential regulators is reduced using an information measure [50, 29]. Others choose edges using a threshold on the minimum of the objective function achieved at the parameter estimates [93]. Despite many practical successes, statistical methodology for feature selection in ODE models is an underdeveloped area. As a result, feature selection in ODE-based network reconstruction proceeds through a mixture of experience, convenience, and analogy.

Our approach, which we call Network Reconstruction via Dyanmical Systems or Network Reconstruction via Dynamic Systems (NeRDS), differs from existing ODE-based methods in the following respects. To begin, we model the right-hand-side function using nonparametric, additive models which are both flexible and data-adaptive. Like other approaches, NeRDS employs a gradient-matching procedure, but differs in that the derivatives are estimated using smoothing-splines rather than finite differences. Finally, we define a novel coupling metric to measure the effect of one component on another allowing us to rank potential edges based on their estimated coupling.

The remainder of the chapter proceeds as follows. Section 4.2 provides both an overview and details of our estimation procedure. Next, in section 4.3 we report numerical results on *in silico* data comparing the NeRDS methodology to other ODE-based methods for network reconstruction. In section 4.4, we synthesize the evidence these performance comparisons provide on the distinct roles of linearity and sparsity, discuss the tradeoffs that accompany the flexibility of the method presented, and point to directions for future work.

4.2 Methods



Stage 1
Time Series Data
Normalize
Smooth
Estimated Trajectories
Estimated Derivatives
Stage 2
Estimated Trajectories
Estimated Derivatives
Select Tuning Parameters by GCV
Fit Additive ODEs via Sparse Backfitting
Estimated Additive Functions Mapping Trajectories to Derivatives
Stage 3
Estimated Additive Functions Mapping Trajectories to Derivatives
Coupling Metrics
Ranked List of Edges

Figure 4.1: *Schematic overview of the NeRDS workflow.* The workflow is split into three stages. The *first stage* involves normalizing and smoothing the data to obtain estimates of the trajectories and derivatives. In this stage each component within each experimental run is smoothed separately. In the *second stage*, for each component an additive model expressing the derivative function in terms of the trajectory functions is fit using the first stage estimates. In the second stage information is combined across experiments, but the models for each component remain separate. Finally, the *third stage* computes pairwise couplings between components to yield a single ranked list of potential edges. FIGURE 4.2 provides a more detailed graphical overview, while the full details of each stage can be found in the Methods section.

4.2.1 Overview

Consider time-course data $(Y^r(t_k), k = 1, ..., n)$ from experiments $r = 1, ..., R$. Suppose the system of interest has d components (metabolites/genes) so that there are also d nodes in the network to be reconstructed. Thus, each $Y^r(t_k) \in \mathbb{R}^d$ is a (random) vector of observations on these d components at time t_k. The data are taken to be noisy observations of an underlying dynamic system,

$$\begin{cases} \dot{x}^r(t) = f(x^r(t)) + u^r(t); \quad x^r(0) = x_0^r, & \text{Process Model} \\ Y^r(t_k) = x^r(t_k) + \epsilon_k^r, & \text{Observation Model.} \end{cases} \tag{4.2}$$

The (known) inputs $u^r(t)$ and the (possibly unknown) initial conditions x_0^r are assumed to vary across experiments so that each trajectory is independently informative of the underlying dynamics. Finally, the measurement errors $\{\epsilon_k^r; k = 1, ..., n; r = 1, ..., R\}$ are assumed to be independent, but not (necessarily) identically distributed.

We take a nonparametric approach in which the right-hand-side function f is subject only to smoothness conditions with $f \in \mathcal{C}^2$ and the second derivative having bounded L_2 norm (see below for additional details). In contrast, other ODE-based methods treat f as known up to some parameters—often assuming a linear or sigmoidal function. The authors in [1] also model f nonparametrically but their approach differs from ours in other respects. Modeling f nonparametrically allows the model to adapt to arbitrary (smooth) nonlinear functions and offers robustness against model mis-specification. However, this also increases the difficulty of the estimation problem. A useful compromise for managing this trade-off is to assume that f is additive so that each component is decomposed as the sum of d univariate functions,

$$f_i(x) = \sum_{i=1}^{d} f_{ij}(x_j). \tag{4.3}$$

Since we do not expect the network to contain edges from all d nodes to node i, the method allows for these additive models to be sparse in the sense that for each i several of the f_{ij} may be equivalently zero. With the additive structure in place, we can state the smoothness conditions precisely as follows. Each of the univariate functions f_{ij} is assumed to belong to the Sobolev space $W_2^2[0,1]$ consisting of twice differentiable functions such that f and $\dot{f}$ are continuous while $\int_0^1 [\ddot{f}(x)]^2 dx < \infty$.

A second feature of the NeRDS methodology is its use of a gradient-matching approach for fitting an ODE to the data. This approach is a straightforward extension of the parametric methods in [32, 7]. One challenge in ODE estimation comes from the fact that while the trajectories are directly observed with error, only indirect information is available on the derivatives. Traditionally this has necessitated computationally expensive numerical integration at each step in the optimization procedure used for parameter estimation.

Gradient-matching approaches avoid this difficulty by first estimating the derivatives on the left-hand side of (4.2) and then using these plug-in estimates to simplify the parameter estimation. This not only avoids costly numerical integration, but also decouples the parameter estimation allowing each component in (4.2) to be learned separately. While gradient-matching approaches have a long history in applied work [20, 13], theoretical guarantees on their performance are quite recent [32].

Most ODE approaches to reconstructing regulatory networks take a gradient-matching approach in which the derivatives are estimated using finite difference approximations [5]. However, in the presence of measurement noise, derivative estimates based on finite differences are inefficient compared to smoothing-based estimates. Smoothing also allows us to estimate the entire derivative, making use of the implicit information between the observation times [32]. Moreover, the additional assumption of smoothness of the underlying trajectories (specifically the continuity of $\ddot{x}$) is not overly cumbersome considering the smoothness required of f, and hence of $\dot{x}$, needed to ensure the existence of a unique solution to the initial value problem (4.2) [31]. By using smoothing splines, we improve on existing gradient-matching procedures in the network reconstruction literature by leveraging smoothness to estimate the derivatives more efficiently.

To summarize, NeRDS is a nonparametric gradient-matching procedure consisting of three stages: normalize and smooth, fit an additive ODE, and estimate coupling metrics. Details of each stage appear below.

4.2.2 Details of the Estimation Procedure

In this section we supply details for a gradient-matching procedure for estimating the right-hand-side function f nonparametrically and using this estimate for network reconstruction. This procedure consists of three stages: 1) smoothing; 2) fitting and additive ODE; and 3) using the estimated ODE to compute the coupling between each pair of nodes. See FIGURE 4.1 for a schematic overview, FIGURE 4.2 for a graphical overview, and ALGORITHM 8 for a high-level description in pseudo-code.

Briefly, the three stages are as follows. In the first stage, we normalize the data and then smooth using splines to obtain estimates of the trajectories and derivatives. Normalization is done within each component across experiments, while smoothing treats each of the d components and R experiments separately so that there are dR distinct smoothing problems to be solved. The second stage consists of solving d additive regression problems treating the estimated derivatives $\hat{\dot{x}}_i$ as response variables and the smoothed trajectories $\hat{x}_i$ as predictors. As with other approaches based on regression, this limits attention to marginal relationships to avoid the combinatorial explosion that would otherwise occur as the number of system components d grows. Finally, in the third stage we compute the pairwise couplings using a normalized version of the L_2 norm of the estimated functions. These couplings allow for ranking potential edges or estimating a network using a threshold. Due to the modularity of the algorithm, adjustments can be made to stage 1 to account for changes to the measurement model without subsequently affecting stages 2 or 3.

4.2.2.1 Stage 1: Normalize and Smooth

Normalization. We begin by normalizing the data to ensure all components are on the same scale. In standard nonparametric modeling, it is common to scale all

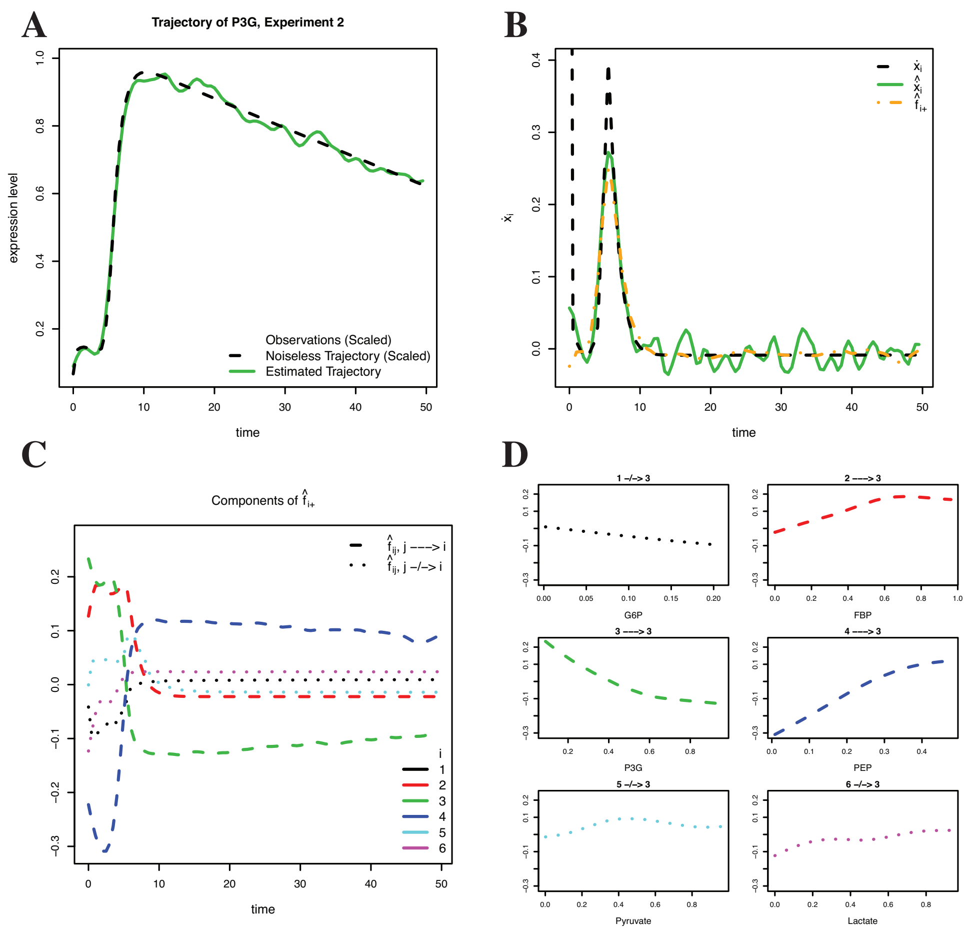


Figure 4.2: *Methodological overview.* This panel illustrates the high-level steps in the NeRDS methodology. **Panel A** shows (simulated) data $(Y_i^r(t_k))_{k=1}^n$ (filled circles) that are noisy measurements of the underlying trajectory (dashed black line) for P3G (i=3) in experiment 2 (r=2). *Step 1:* Smooth the data to estimate the trajectory (solid green line). **Panel B** *Step 2:* Estimate the derivative (dashed black line) using the derivative of the estimated trajectory (solid green line). *Step 3:* Aggregating estimates across experiments, fit an additive nonparametric function (dot-dash orange line) expressing the estimated derivative in terms of the estimated trajectories. **Panel C** shows the components of the additive function, each of which is a univariate function of a single trajectory. Specifically shown here are $\{\hat{f}_{3j}(\hat{x}_j^2(t))\}_{j=1}^6$ plotted against time. **Panel D** shows the component functions $\{\hat{f}_{3j}\}_{j=1}^x$ plotted over their domain (i.e. $[\min_{t,r} x_j^r(t), \max_{t,r} x_j^r(t)]$). *Step 4:* Estimate couplings using an L_2 norm of the estimated component functions. In panels C and D, solid lines inidcated regulators of P3G in the underlying network and dashed lines non-regulators.

variables to have standard deviation one. This makes the resulting models invariant to scale and allows regularization to proceed without additional weighting schemes. However, since our observations are functional, we scale instead by the maximum observed value for each component across experiments. This serves a similar purpose and is also the approach taken in the DREAM competitions [53].

Using smoothing to estimate the trajectories and derivatives. The purpose of the first stage is to obtain estimated time derivatives, $\hat{\dot{x}}_i(t)$; smoothed trajectory estimates $\hat{x}_i(t)$ are a welcome byproduct. Beginning with the trajectories, for each component i and experiment r the estimated trajectories satisfy,

$$\hat{x}_i^r(t) = \arg \min_{x \in W_2^2[0,1]} \sum_{k=1}^{n} [y_i^r(t_k) - x(t_k)]^2 + \lambda_{0,i}^r \int_0^1 [\ddot{x}(t)]^2 dt, \tag{4.4}$$

where $y_i^r(t_k)$ is the (normalized) observation of component i at time t_k in experiment r and $W_2^2[0,1]$ is the Sobolev space discussed in the overview. The solution is a natural cubic spline with knots at the unique time points [28]. The estimated trajectories are given by the basis function expansion $\hat{x}_i^r(t) = b(t)\hat{\gamma}_i^r$ where $b(t) = (b_1(t), ..., b_n(t))$ is the (row) vector of smoothing-spline basis functions evaluated at time t and $\hat{\gamma}_i^r$ are the coefficients solving a finite-dimensional version of (4.4),

$$\hat{\gamma}_i^r = \arg \min_{\gamma \in \mathbb{R}^n} \sum_{k=1}^{n} [y_i^r(t_k) - b(t_k)\gamma]^2 + \gamma'\Omega\gamma, \qquad \Omega_{\ell m} = \int_0^1 \ddot{b}_\ell(t)\ddot{b}_m(t)dt. \tag{4.5}$$

Derivatives estimates are obtained by differentiating the estimated trajectories,

$$\hat{\dot{x}}_i^r(t) = \frac{d}{dt}\hat{x}_i^r(t) = \dot{b}(t)\hat{\gamma}_i^r. \tag{4.6}$$

Both the trajectories and derivatives are easily computed using standard software which also allows for efficient estimation of tuning parameters $\{\lambda_{i0}^r; i = 1, ..., d; r =$

$1, ..., R\}$ by cross validation or generalized cross validation.

4.2.2.2 Stage 2: Fit an Additive ODE

The second stage involves finding an additive nonparametric model relating the estimated derivatives $\hat{\dot{x}}(t)$ to the estimated trajectories $\hat{x}(t)$. Specifically, if $u_i^r(t) = 0$ the second stage minimizes,

$$\hat{M}_{n,r}(f) = \int_0^1 \left[\hat{\dot{x}}_i^r(t) - \sum_{j=1}^d f_{ij}(\hat{x}_j^r(t)) \right]^2 dt + \lambda_{1i} \sum_{j=1}^d \int_{\mathcal{R}_j} [\ddot{f}_{ij}(x)]^2 dx, \tag{4.7}$$

so that the estimator is,

$$\hat{f}_i = \arg\min_{f_i \in \mathcal{D}} R^{-1} \sum_{r=1}^R \hat{M}_{n,r}(f), \tag{4.8}$$

with $\mathcal{D} = \{f : f = \sum_{j=1}^d f_j, f_j \in W_2^2[\mathcal{R}_j]\}$ and $\mathcal{R}_j = [\min_{t,r} \hat{x}_j^r(t), \max_{t,r} \hat{x}_j^r(t)]$ an interval covering the estimated range of component j over all experiments. If $u_i^r(t) \neq 0$ it should be subtracted from $\hat{\dot{x}}_i(t)$ before solving the optimization problem above.

The objective function (4.7) is minimized using sparse backfitting [74], see ALGORITHM S2. Backfitting is a technique for fitting nonparametric additive models by iteratively applying univariate smoothers [35, 8]. In our case this involves first centering the estimated derivatives about the component mean and then successively solving univariate smoothing-spline problems. For instance, to update the j^{th} component in the i^{th} model, solve,

$$\breve{f}_{ij} = \arg\min_{f \in W_2^2[\mathcal{R}_j]} R^{-1} \sum_{r=1}^R \int_0^1 \left[\left(\tilde{\dot{x}}_i^r(t) - \sum_{\ell \neq j} \tilde{f}_{i\ell}(\hat{x}_\ell^r(t)) \right) - f(\hat{x}_\ell^r(t)) \right]^2 dt + \lambda_{i1} \int_{\mathcal{R}_j} [\ddot{f}(x)]^2 dx, \tag{4.9}$$

where $\tilde{f}_{i\ell}$ are the current estimates and $\tilde{\dot{x}}_i^r$ are the centered derivatives. In practice, the integrals are approximated using quadrature and the above can be accomplished by premultiplying the residual vector by a smoothing matrix S_j. Although this is a useful simplification, rather than premultiplying by S_j we solve the corresponding linear system using a QR decomposition to obtain updated estimates of the basis expansion coefficients γ_i. The $\breve{f}_{ij}$ are next centered for identifiability. Finally, in order to induce sparsity, a soft-threshold is applied after solving the smoothing problem, so that the update is,

$$\tilde{f}_{ij} = (1 - \lambda_{i2}/||\breve{f}||_2)_+ \breve{f}_{ij}. \tag{4.10}$$

4.2.2.3 Tuning the smoothing and sparsity parameters

The estimators $\hat{f}_i$ from stage two depend on tuning parameters $\lambda_i = (\lambda_{i1}, \lambda_{i2})$. These tuning parameters depend on i because each of the d submodels is fit separately. The smoothing parameter could be allowed to vary by component, $\lambda_{i1} = (\lambda_{i1j})_{j=1}^d$, but at the cost of greatly expanding the computational cost required for tuning. In our experience this additional flexibility does not lead to significant improvement in terms of network reconstruction. The smoothing parameter λ_{i1} controls the roughness of the individual functions f_{ij} while the sparsity parameter λ_{i2} induces sparsity by setting some of the f_{ij} to zero. These tuning parameters are selected by minimizing the Generalized Cross Validation (GCV) score suggested by [74],

$$GCV(\lambda_i) = \frac{(nR)^{-1} \sum_{r=1}^R \sum_{k=1}^n [\hat{\dot{x}}_i^r(t_k) - \hat{f}_i(\hat{x}^r(t_k); \lambda_i)]^2}{(1 - df(\lambda_i)/n)^2}, \tag{4.11}$$

where $df(\lambda_i) = \sum_{j=1}^d df_j(\lambda_i) 1[f_{ij} \neq 0]$ and $df_j(\lambda_i) = tr(B_j(B_j'B_j + \lambda_{i1}\Omega_j)^{-1}B_j')$ is the trace of the hat matrix projecting onto the span of the b-spline basis for the j^{th} component.

While GCV allows for automatic selection of tuning parameters, overfitting—selecting λ_1 or λ_2 too small so the resulting model is overly complex—is always a concern. In fact, it is our experience from simulation studies that overfitting is the norm when using GCV with our methodology. As a first pass, one may choose to select λ_1 fairly large, say, $\lambda_1 = 1$, or reduce the number of knots employed, so that the resulting additive functions are nearly linear. One can then decrease λ_1 toward the value selected by GCV or increase the number of knots until an appropriate balance between flexibility and complexity is achieved, with the 'appropriate' balance depending on context.

Likewise, the search range for λ_2 should be chosen large enough to ensure convergence of sparse backfitting in a reasonable number of iterations, yet small enough to ensure a meaningful model. Within this range GCV can serve as an objective guideline from which to justify specific departures.

Model diagnostics are an important tool for balancing complexity and flexibility. Plots overlaying estimated derivatives with linear and selected additive fits can be used to discover places where the additional flexibility is needed to achieve an adequate fit or where the complexity can be restricted without undue loss of fit. In section C, we illustrate use of these diagnostics for select terms from the mouse system explored in section 4.3.

4.2.2.4 Identifiability Issues

Given the complexity of the model class, it is natural to wonder about the identifiability of the additive model. To this end it is relevant to note that the smoothing matrices, S_j used to solve (4.7), are symmetric linear smoothers with eigenvalues in [0,1]. Hence, the backfitting procedure will converge to a minimizer of (4.7) (cf. [35], pg. 122).

However, this minimizer need not be unique despite the identifiability requirement $\int_{\mathcal{R}_j} \hat{f}_{ij}(x)dx = 1$. The uniqueness will depend on the *concurvity* space of the smoothers [35]. Namely, let $\mathcal{M}_1(S_j)$ be the space spanned by the first eigenvector of

S_j. Concurvity can be thought of as the functional analog to collinearity. Then the concurvity space is,

$$\mathcal{M} = \{(g_1, ..., g_d) : g_j \in \mathcal{M}_1(S_j), \sum_{j=1}^{d} g_j = 0, \}/(0, ..., 0). \tag{4.12}$$

If $\mathcal{M}$ is empty then the solution to (4.7) will be unique. If not, the backfitting algorithm will still converge, but the solution will depend on the initial estimates of the f_{ij}.

In practice, we computationally check the identifiability of our fitted model in the following way. Since we always initialize at $f_{ij} \equiv 0$, the initial estimates of the f_{ij} depend on the order in which the backfitting is carried out. Thus, to check for identifiability we permute this order a number of times (say 10) and compute the resulting backfitting estimators, $\hat{f}^a_{ij}, a = 1, ..., 10$. We then compute pairwise L_∞ distances between the estimates,

$$d_{ab} = \sup_{x \in \mathcal{R}_j} |f^a_{ij}(x) - f^b_{ij}(x)|. \tag{4.13}$$

When these distances are of the order of the threshold ϵ used to define convergence we take this as evidence of identifiability. Otherwise, the larger these distances the stronger the evidence against the uniqueness of each fit is. In practical terms, it also helps to overlay plots of the resulting fits and observe the extent to which they agree.

Often, when the model is not identified, it is the result of the data being insufficient for the complexity of the model fitted. Hence, reducing this complexity by increasing λ_1 or λ_2 until the model becomes identified is an attractive option that we have had success with. From extensive simulation studies we find that having $R \geq d$, at least as many experiments as system components, generally suffices for identifiability.

4.2.3 Coupling Metrics

The process model in (4.2) is specified by a set of coupled ODEs. The link between the dynamic system (4.2) and the target network is formalized by defining edges based on the relevant variables in the right-hand-side function f. Specifically, component j regulates component i if the i^{th} component of f explicitly depends on x_j,

$$j \to i \iff \frac{\partial f_i}{\partial x_j} \not\equiv 0. \tag{4.14}$$

In the general case, the partial derivative $\frac{\partial f_i}{\partial x_j}(x(t))$ is a function of $x(t)$—the concentrations of all components at time t. Since our working models are additive, $f_{i+} = \sum_{i=1}^{d} f_{ij}$, the partial derivatives $\frac{\partial f_{i+}}{\partial x_j}(x_j)$ depend at most on x_j. Moreover,

$$\frac{\partial f_{i+}}{\partial x_j}(x_j) = 0 \iff f_{ij} \equiv 0, \tag{4.15}$$

allowing us to use the coupling metric,

$$\rho_{ij} := \sqrt{\frac{\int_{\mathcal{R}_j} [\hat{f}_{ij}(z)]^2 dz}{|\mathcal{R}_j|}}, \tag{4.16}$$

with $\mathcal{R}_j$ the observed range of x_j and $|\mathcal{R}_j|$ its length.

The coupling metrics are used to rank potential edges based on the strength of their regulatory influence. If desired, a single estimated network can be obtained by choosing a threshold for the coupling; only edges with coupling above this threshold are included in the estimated network. Strategies for choosing this threshold are a subject of ongoing research.

For recovering signed edges—corresponding to, say, promotion and inhibition—we define signed coupling metrics,

$$\rho_{ij}^{+} := \sqrt{\frac{\int_{\mathcal{R}_j}[(\hat{\dot{f}}_{ij}(z))_{+}]^2 dz}{|\mathcal{R}_j|}}, \qquad \rho_{ij}^{-} := \sqrt{\frac{\int_{\mathcal{R}_j}[(\hat{\dot{f}}_{ij}(z))_{-}]^2 dz}{|\mathcal{R}_j|}}, \tag{4.17}$$

by taking the positive $(\cdot)_{+} = \max\{\cdot, 0\}$ and negative $(\cdot)_{-} = \min\{\cdot, 0\}$ parts, respectively.

4.3 Results

We evaluate the performance of our method *in silico* using simulated data from a variety of computational models for real biological systems. In each case, the computational model is specified by a highly nonlinear ODE with the collection of systems chosen to reflect a representative cross-section of canonical functional forms. Specifically, we choose examples from: the S-system formalism [88], sigmoidal dynamics popular with computational modelers [11], as well as the thermodynamics-based models used in the DREAM competitions [53].

Within each system, we apply the NeRDS methodology for estimating the coupling by constructing a nonparametric additive ODE and compare it to three standard parametric alternatives: linear ODEs, linear ODEs plus ℓ_1 regularization (Lasso), and Inferelator 1.0 [5]. Inferelator 1.0 also employs linear ODEs and the Lasso, but takes a slightly different approach to estimation. The key differences are: 1) estimation of derivatives by finite differencing rather than smoothing splines; 2) construction of a response variable for each node combining the estimated derivative and trajectory at each time point; and 3) use of the raw observations rather than smoothed estimates of trajectories as covariates. All methods in the comparisons employ gradient-matching and use (misspecified) ODE models to estimate the network connections from time course data.

The models resulting from each method are used to rank potential edges in terms of their coupling. For the linear models, the estimated coupling is simply the appropriate coefficient in the transfer matrix. The methods' utility for network reconstruction are then compared in terms of AUC-ROC and AUC-PR. In sparse models the order of potential edges at the bottom of the ranked lists (corresponding to zero estimated coupling) is arbitrary. To account for this we approximate the expected AUC under random orderings of the remaining edges.

4.3.1 Metabolic Pathway in Lactocaccus Lactis

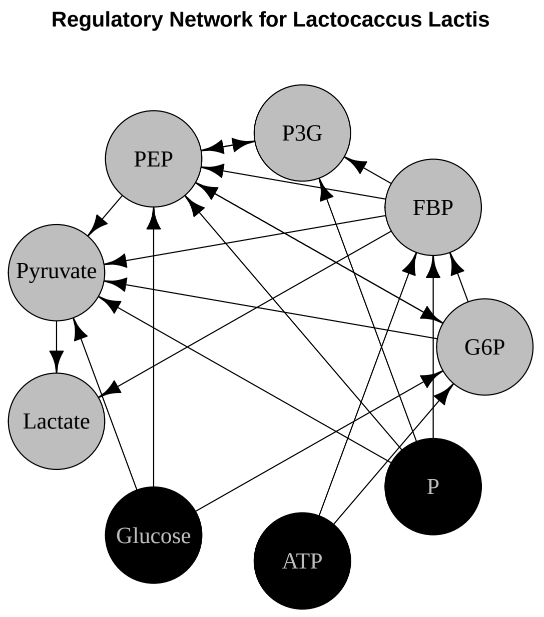


Figure 4.3: *Network topology for the Lactocaccus Lactis system.* The dark nodes with light text (glucose, ATP, and phosphorus) correspond to offline variables not explicitly modeled. We focus on reconstructing the subnetwork among the online variables (light nodes with dark text). The (simulated) data consists of vector-valued time-series of the metabolites represented by the nodes. The network, computational model and data for offline variables are taken from [89].

We begin by evaluating our methodology on an S-system developed to describe the conversion of glucose to lactate via an Embden-Meyerhof glycolytic pathway in the

Lactococcus Lactis bacterium [89]. The system consists of nine metabolites of which three (glucose, ATP, and phosphate) are offline variables not explicitly modeled. See FIGURE 4.3 for the network topology. For evaluation purposes we aim to reconstruct only the subnetwork among the six online variables for which the network formalism (4.14) makes sense.

Data for the reconstruction were obtained by simulating a suite of six experiments using the model in [89]. This suite was designed to induce curvature in the trajectories sufficient to make the nonparametric additive model identifiable. Moreover, the experiments compliment one another by ramping up the coupling among targeted subsets of edges. Specifically, this was accomplished by altering the initial abundance of each metabolite in turn,

$$\begin{cases} x_i^r(0) = x_{0i}, & i \neq r \\ x_i^r(0) = Mx_{0i}, & i = r. \end{cases} \tag{4.18}$$

The magnitude, M, of the simulated perturbations is a simulation parameter loosely corresponding to how substantially the six experiments differ from one another.

Noiseless trajectories for each simulated experiment were computed via numerical integration. The trajectories were sampled at $n = 100$ times $\{t_k = \frac{k-1}{n}49, k = 1, ..., n\}$ with noise added to simulate measurement error,

$$Y^r(t_k) = x^r(t_k) + \epsilon_{rk}, \quad \epsilon_{ki}^r \overset{indp.}{\sim} N(0, \sigma x_i^r(t_k)). \tag{4.19}$$

We carried out simulations for $\sigma \in \{.02, .05\}$ and $M \in \{15, 10, 5, 2, 1.5\}$ with 500 repetitions for each (σ, M) pair. The simulated data from each repetition were normalized as described in the Methods section and then used as input to four network reconstruction algorithms: 1) a NeRDS additive ODE with four interior knots, $\lambda_2 = 0$ and λ_1 selected by GCV searching over the grid $\{.05z, z = 1, ..., 10\}$; 2) a linear ODE fit by gradient matching; 3) a sparse linear ODE fit using gradient matching and lars [34]; and 4) Inferelator 1.0 [5]. All simulations were done using `R` [72].

Each of the first three methods utilize smoothing splines to smooth the trajectories and estimate the derivatives, as described in Methods. For the additive ODEs the sparsity parameter λ_2 was set to 0 due to the small size of the system while the number of knots and search-range for λ_1 were selected by examining diagnostic plots as discussed in Methods. Moreover, following the estimation of ODE parameters using each algorithm, we ranked potential edges using the coupling metric introduced in Methods. For the three approaches employing linear ODEs, this reduces to ranking edges by the magnitude of estimated entries in the transfer matrix.

The mean AUC-PR and AUC-ROC from the *Lactocaccus* simulations appear in TABLE 4.1 and TABLE 4.3.1, respectively. The dispersion of these measures among the 500 repetitions can be seen in the box plots of FIGURE 4.4. Additional results using reduced sampling densities, $n \in \{25, 50, 75\}$, appear in FIGURE C.3. In terms of AUC-ROC , the additive ODEs used by NeRDS outperformed the competitors with the exception of low-signal ($M = 2, 1.5$) high-noise ($\sigma = .05$) settings. Evaluated on the basis of precision-recall scores the additive ODEs performed best in high-signal settings ($M \geq 5$), but dropped off considerably under more modest perturbations. Taken together, these results indicate that moving from linear to additive ODEs takes better advantage of sufficiently strong signals. In low-signal settings ($M \leq 2$) focused on precision-recall, the sparse methods, linear ODEs + Lasso and Inferelator, outperformed the methods not utilizing sparsity.

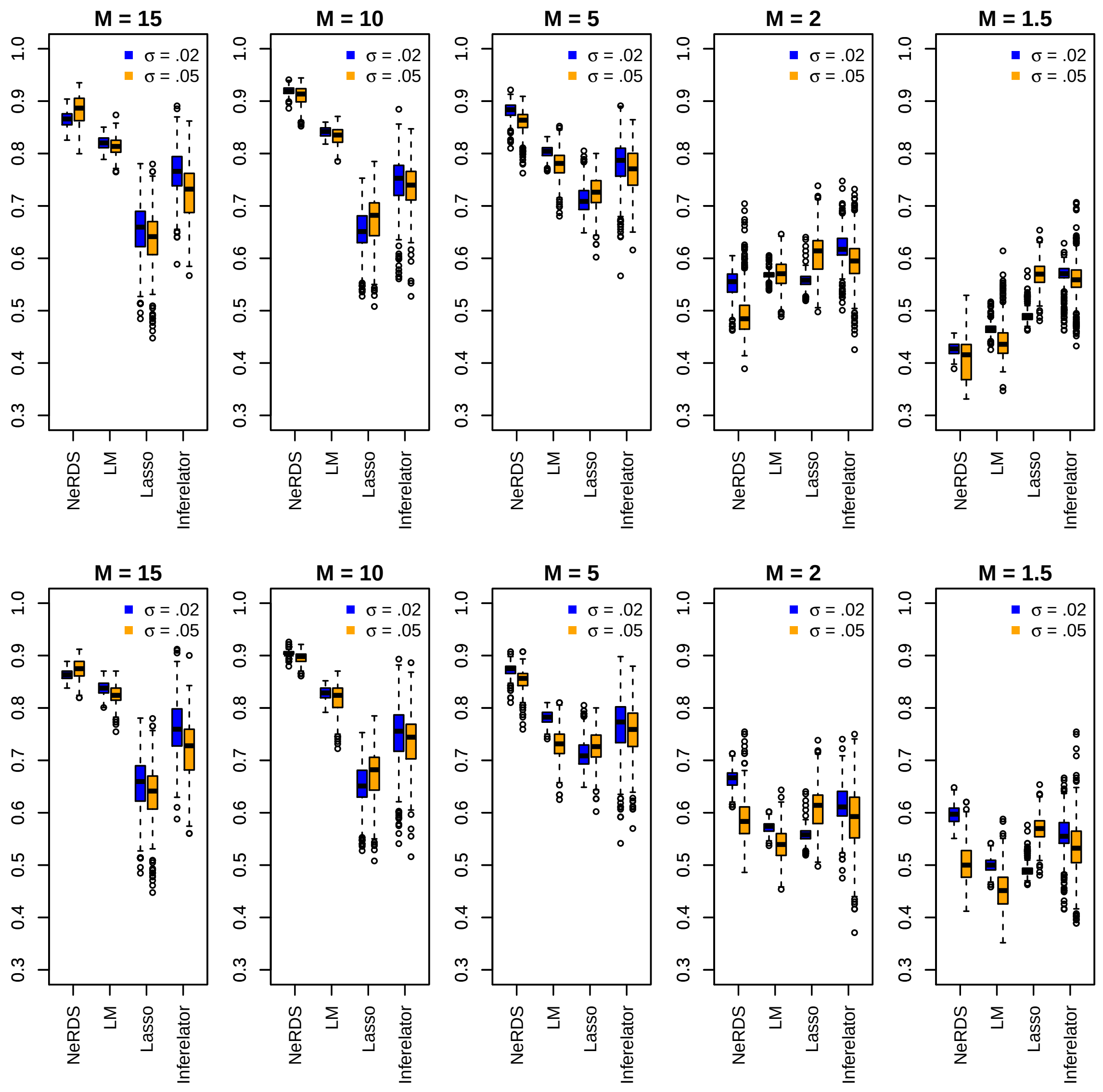


Figure 4.4: *Performance evaluation on the Lactococcus Lactis system. Upper row:* Box Plots showing area under the precision-recall curves from 500 Monte Carlo simulations reconstructing the *Lactococcus* network. *Bottom Row:* Box Plots showing area under the ROC curves. Each plot in the row corresponds to a different value of the perturbation parameter M (high to low from left to right). Within each plot, box plots are arranged according to reconstruction method; from left to right these are: additive ODEs (NeRDS), linear ODEs (LM), linear ODEs plus a Lasso penalty (Lasso), and Inferelator. For each method a pair of box plots are presented corresponding to low noise ($\sigma = 0.02$, left) and moderate noise ($\sigma = .05$, right).

	$\sigma = .02$	$\sigma = .05$
M=15, Additive ODE	**.87** (.865, .867)	**.88** (.881, .886)
M=15, Linear ODE	.82 (.819, .821)	.81 (.812, .815)
M=15, Linear ODE + Lasso	.65 (.650, .659)	.64 (.632, .642)
M=15, Inferelator 1.0	.76 (.761, .769)	.73 (.722, .731)
M=10, Additive ODE	**.92** (.918, .920)	**.91** (.909, .912)
M=10, Linear ODE	.84 (.840, .841)	.83 (.832, .835)
M=10, Linear ODE + Lasso	.65 (.650, .657)	.67 (.669, .677)
M=10, Inferelator 1.0	.75 (.741, .750)	.74 (.734, .741)
M=5, Additive ODE	**.88** (.881, .883)	**.86** (.859, .862)
M=5, Linear ODE	.80 (.802, .804)	.78 (.776, .781)
M=5, Linear ODE + Lasso	.71 (.710, .715)	.73 (.723, .729)
M=5, Inferelator 1.0	.78 (.778, .787)	.77 (.764, .772)
M=2, Additive ODE	.55 (.549, .553)	.49 (.490, .498)
M=2, Linear ODE	.57 (.567, .569)	.57 (.567, .572)
M=2, Linear ODE + Lasso	.56 (.556, .559)	**.61** (.605, .612)
M=2, Inferelator 1.0	**.62** (.618, .624)	.60 (.592, .599)
M=1.5, Additive ODE	.43 (.426, .428)	.41 (.403, .410)
M=1.5, Linear ODE	.47 (.464, .466)	.44 (.439, .445)
M=1.5, Linear ODE + Lasso	.49 (.490, .493)	**.57** (.568, .572)
M=1.5, Inferelator 1.0	**.57** (.563, .568)	.56 (.556, .562)

Table 4.1: *Area under the precision-recall curve for the Lactococcus Lactis system.* Performance comparison in terms of area under the precision recall curve of four methods for reconstructing the *Lactococcus* network. The figures given are averages from 500 Monte Carlo repetitions along with confidence intervals for the mean. The parameter M corresponds to the size of the perturbation used in generating the time series while the standard deviation of the noise is proportional to σ. Six time series, each with $n = 100$ observations, are used in the reconstruction.

	$\sigma = .02$	$\sigma = .05$
M=15, Additive ODE	**.86** (.863, .864)	**.88** (.874, .877)
M=15, Linear ODE	.84 (.836, .838)	.82 (.822, .825)
M=15, Linear ODE + Lasso	.65 (.650, .659)	.64 (.632, .642)
M=15, Inferelator 1.0	.76 (.755, .764)	.72 (.716, .727)
M=10, Additive ODE	**.91** (.904, .906)	**.90** (.895, .897)
M=10, Linear ODE	.83 (.826, .828)	.82 (.815, .820)
M=10, Linear ODE + Lasso	.65 (.650, .657)	.67 (.669, .677)
M=10, Inferelator 1.0	.75 (.744, .753)	.74 (.733, .742)
M=5, Additive ODE	**.87** (.871, .874)	**.85** (.852, .856)
M=5, Linear ODE	.78 (.781, .783)	.73 (.726, .731)
M=5, Linear ODE + Lasso	.71 (.710, .715)	.73 (.723, .729)
M=5, Inferelator 1.0	.77 (.764, .774)	.76 (.751, .759)
M=2, Additive ODE	**.66** (.663, .666)	.59 (.584, .591)
M=2, Linear ODE	.57 (.572, .574)	.54 (.537, .542)
M=2, Linear ODE + Lasso	.56 (.556, .559)	**.61** (.605, .612)
M=2, Inferelator 1.0	.61 (.612, .618)	.59 (.586, .597)
M=1.5, Additive ODE	**.60** (.596, .599)	.50 (.499, .506)
M=1.5, Linear ODE	.50 (.499, .502)	.45 (.450, .457)
M=1.5, Linear ODE + Lasso	.49 (.490, .493)	**.57** (.568, .572)
M=1.5, Inferelator 1.0	.56 (.552, .559)	.54 (.531, .540)

Table 4.2: *Area under the ROC curve for the Lactococcus Lactis system.* Performance evaluation for the *Lactococcus* network using area under the ROC curve.

4.3.2 Gene Regulatory Network in Mouse Embryonic Stem Cells

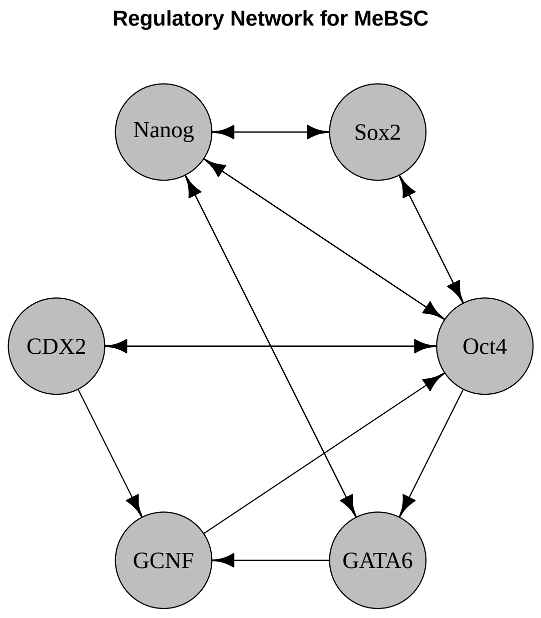


Figure 4.5: *Network topology for the mouse embryonic stem cell system.* This six-gene regulatory network consists of 14 edges (regulatory relationships) which we wish to discover from time-series observations of the gene expressions. The network and the computational model used to generate these observations are taken from [11].

Our second example for evaluating the NeRDS methodology is a computational model for a six-gene regulatory network developed to explain lineage determination of embryonic stem cells in mice [11]. See FIGURE 4.5 for the network topology. The system of ODEs describing the network is based on a thermodynamic model for gene regulation resulting in sigmoidal functional forms involving two- and three-way interaction terms. The setup for the simulations was nearly identical to that used for the *Lactocaccus* system described previously, with the exception that the $n = 100$ observation times $\{t_k = \frac{k-1}{n}30, k = 1, ..., n\}$ span a lesser duration. While the additive ODEs again used four interior knots and fixed $\lambda_2 = 0$, λ_1 was chosen by GCV searching over the grid $\{10^z, z = -2, -1.5, -1\}$. The number of knots and search range were selected to be as close to linear as possible while providing adequate fit as assessed by examining diagnostic plots from a representative dataset. See FIGURE C.1 for an example and section C for more on this point.

Simulation results for reconstructing the mouse network appear in TABLE 4.3.2 and TABLE 4.4, showing mean areas under, respectively, the precision recall and ROC curves from 500 repetitions at a variety of settings. The results are also presented graphically using box plots in FIGURE 4.6 giving a sense of each method's variability. Performance using reduced sampling densities, $n \in \{25, 50, 75\}$, can be found in FIGURE C.4.

For this network, the additive and linear ODEs were clearly the best performers overall. As with the *Lactocaccus* network, additive ODEs were the best performers in high-signal settings ($M \geq 5$). Linear ODEs had a slight advantage in low-signal ($M \leq 2$) high-noise ($\sigma = .05$) settings, while the two methods are virtually indistinguishable in low-signal ($M \leq 2$), low-noise ($\sigma = .02$) settings. Looking at the boxlots in FIGURE 4.6, we see that in low-signal ($M \leq 2$), high-noise ($\sigma = .05$) settings additive and linear ODEs both occasionally achieved perfect reconstructions, but that linear ODEs performed slightly better on average due by having higher worst-case performance.

To some extent the lack of robustness displayed may be an artifact of the simulation setting as the need to do 500 Monte Carlo repetitions precluded us from checking the stability of each model as discussed in the section Identifiability Issues above. In practice, these stability checks should suggest higher values of λ_1 so that the additive ODEs of NeRDS become more similar to their linear counterparts. This also suggests that, all else equal, users of NeRDS should favor higher values of the smoothness parameter λ_1 and also consider using fewer knots; see the discussion section for more on this point.

Both linear and additive ODEs outperformed Inferelator as did linear ODEs +

Lasso implying that this difference can not be attributed solely to sparsity. The observed differences between linear ODE + Lasso and Inferelator likely reflect the additional stability of the former due to the way in which the derivatives are estimated; in most cases, smoothing splines provide better derivative estimates than finite differencing. Nevertheless, sparsity clearly did play a role as the two methods not employing sparsity (NeRDS and linear ODEs) performed better than those that did (linear ODEs + Lasso and Inferelator). Note that NeRDS did not employ sparsity because we fixed $\lambda_2 = 0$.

	$\sigma = .02$	$\sigma = .05$
M=15, Additive ODE	**.96** (.961, .962)	**.96** (.958, .960)
M=15, Linear ODE	.96 (.959, .960)	.95 (.944, .948)
M=15, Linear ODE + Lasso	.74 (.742, .746)	.74 (.739, .743)
M=15, Inferelator 1.0	.65 (.640, .654)	.61 (.604, .618)
M=10, Additive ODE	**.98** (.980, .981)	**.98** (.977, .978)
M=10, Linear ODE	.96 (.963, .963)	.96 (.953, .957)
M=10, Linear ODE + Lasso	.75 (.744, .746)	.74 (.736, .741)
M=10, Inferelator 1.0	.66 (.655, .668)	.62 (.615, .629)
M=5, Additive ODE	**.98** (.984, .985)	**.98** (.979, .981)
M=5, Linear ODE	.97 (.969, .970)	.96 (.963, .965)
M=5, Linear ODE + Lasso	.75 (.751, .753)	.74 (.740, .745)
M=5, Inferelator 1.0	.70 (.696, .708)	.65 (.641, .656)
M=2, Additive ODE	**.98** (.977, .979)	.94 (.935, .941)
M=2, Linear ODE	**.98** (.976, .978)	**.96** (.953, .958)
M=2, Linear ODE + Lasso	.76 (.758, .762)	.74 (.741, .748)
M=2, Inferelator 1.0	.70 (.700, .707)	.61 (.601, .614)
M=1.5, Additive ODE	.97 (.966, .970)	.88 (.873, .883)
M=1.5, Linear ODE	**.97** (.971, .974)	**.90** (.899, .908)
M=1.5, Linear ODE + Lasso	.76 (.757, .763)	.73 (.730, .740)
M=1.5, Inferelator 1.0	.66 (.651, .661)	.55 (.548, .562)

Table 4.3: *Area under the precision-recall curve for the mouse system.* Performance comparison using area under the precision-recall curve for reconstructing the mouse network.

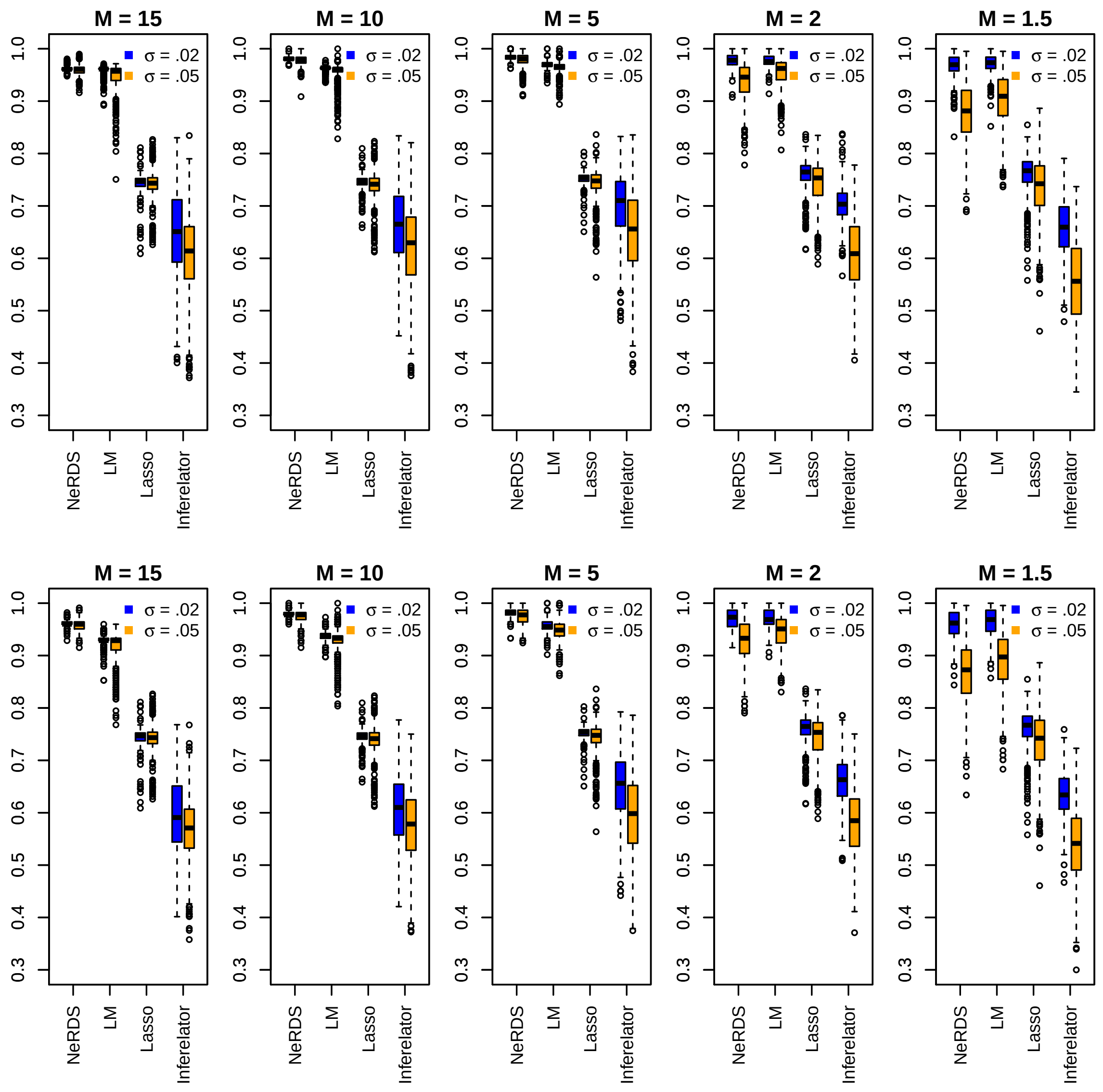


Figure 4.6: *Performance evaluation on the mouse embryonic stem cell system. Upper row:* Box Plots showing area under the precision recall curves from 500 Monte Carlo simulations reconstructing the Mouse network. *Bottom Row:* Box Plots showing area under the ROC curves. Each plot in the row corresponds to a different value of the perturbation parameter M (high to low from left to right). Within each plot, box plots are arranged according to reconstruction method; from left to right these are: additive ODEs (NeRDS), linear ODEs (LM), linear ODEs plus a lasso penalty (Lasso), and Inferelator. For each method a pair of box plots are presented corresponding to low noise ($\sigma = 0.02$, left) and moderate noise ($\sigma = .05$, right).

	$\sigma = .02$	$\sigma = .05$
M=15, Additive ODE	**.96** (.961, .962)	**.96** (.957, .959)
M=15, Linear ODE	.93 (.929, .930)	.92 (.914, .919)
M=15, Linear ODE + Lasso	.74 (.742, .746)	.74 (.739, .743)
M=15, Inferelator 1.0	.59 (.587, .600)	.56 (.559, .570)
M=10, Additive ODE	**.98** (.979, .980)	**.98** (.974, .976)
M=10, Linear ODE	.94 (.936, .938)	.93 (.926, .930)
M=10, Linear ODE + Lasso	.75 (.744, .746)	.74 (.736, .741)
M=10, Inferelator 1.0	.60 (.598, .611)	.57 (.567, .579)
M=5, Additive ODE	**.98** (.982, .983)	**.98** (.975, .977)
M=5, Linear ODE	.96 (.956, .958)	.95 (.946, .949)
M=5, Linear ODE + Lasso	.75 (.751, .753)	.74 (.740, .745)
M=5, Inferelator 1.0	.65 (.644, .655)	.60 (.588, .602)
M=2, Additive ODE	**.97** (.969, .972)	.93 (.925, .932)
M=2, Linear ODE	**.97** (.968, .971)	**.95** (.943, .949)
M=2, Linear ODE + Lasso	.76 (.758, .762)	.74 (.741, .748)
M=2, Inferelator 1.0	.66 (.658, .665)	.58 (.577, .589)
M=1.5, Additive ODE	**.96** (.958, .962)	.87 (.861, .872)
M=1.5, Linear ODE	**.96** (.962, .967)	**.89** (.886, .896)
M=1.5, Linear ODE + Lasso	.76 (.757, .763)	.73 (.730, .740)
M=1.5, Inferelator 1.0	.63 (.630, .638)	.54 (.534, .547)

Table 4.4: *Area under the ROC curve for the mouse system.* Performance comparison using area under the ROC curve for reconstructing the Mouse network.

4.3.3 DREAM 3 10- and 100-Node Networks

In addition to the computational models described above, we also evaluated the NeRDS methodology on the 10- and 100-node networks from DREAM 3, challenge 4 [53, 52, 70]. This provides performance comparisons on a premier evaluation model and including the 100-node networks allows us to demonstrate that NeRDS is scalable despite its complexity relative to parametric models.

While the DREAM 3 networks represent an important point of comparison an observation is in order. Unlike the *Lactocaccus* and Mouse examples in which the time evolution of the system is fully observed, the DREAM 3 dynamics are only partially observed. This is due to the dynamic system generating the data involving unobserved proteins. The presence of unobserved variables adds an additional layer of approximation for the working models to accomodate. Including unobserved variables in the generating model has the advantage of being more faithful to the underlying science but makes *de novo* exploration more difficult. For this reason, we should not expect general exploratory models, such as linear ODEs or the additive nonparametric ODEs used by NeRDS, to perform as well as methods that take full advantage of prior scientific knowledge.

We used GeneNetWeaver [52] to generate, respectively, 10 and 100 multifactorial time series for each of the five DREAM 3 10- and 100-node networks. As discussed in [53], GeneNetWeaver generates multifactorial time series by integrating the *in silico* model from various initial conditions. These multifactorial time series are meant to simulate the networks' response to global perturbations. Here 'global' signifies that the targets of the perturbations are unknown, so that one can neither employ direct cause-effect methodologies, nor incorporate such cause-effect information into a dynamic model. Similar to the competition settings, these time series were generated using ODEs and adding Gaussian measurement error with standard deviation .025 to the $n = 21$ observation times on each time-series. As before, we applied each of the four methods under consideration to reconstruct these networks on the basis of these time series. We look at mean performance in terms of AUC-PR and AUC-ROC over 500 realizations of the measurement error for the 10-node networks, and 10 realizations of the noise for each 100-node network. The results, displayed in Table 4.5 and Table 4.6 indicate that the additive ODEs we employ compare favorably with other methods.

For the additive ODEs on the 100-node networks, we first computed GCV over a range of λ_1 and λ_2 values for a single repetition from the Ecoli1 100-node network with knots at all unique data points. To improve stability and limit complexity, we fixed $\lambda_1 = .1$ and $\lambda_2 = .001$ across all nodes in the 100-node networks because these values most frequently minimized GCV on the network examined. This has the effect of eliminating variability due to tuning parameter selection. For the 10-node networks, we used 4 knots and fixed $\lambda_2 = 0$, but allowed λ_1 to be selected by GCV from the sequence $\lambda_1 \in \{10^z, z = -1, -.8, -.6, ..., .6, .8, 1\}$.

Unlike the DREAM 3 competition, in the comparison just discussed we did not assume access to any knockdown or knockout data in accordance with our goal of improving methodology for time-course data. However, for the sake of completeness, we also provide performance comparisons on the actual data from challenge 4 of the DREAM 3 competition. Due to the small number of time series available (4 and 46, respectively, for the 10- and 100-node networks), methods not utilizing the knockout data—known to be most informative [53]—will not be competitive. For a fair comparison, we first used the knockout data to estimate an influence matrix for each network. Using this estimated influence matrix, we limited the pool of potential regulators for each submodel when fitting additive ODEs to the time series data. In summary, we screened potential regulators using the knockout experiments, and then ranked those remaining in terms of the estimated coupling.

Our approach to estimating the influence matrix was similar to that used by the top performers in the competition for estimating the first batch of edges [93]. Briefly, the idea is to use t-tests to determine which genes in a particular knockout strain have expression levels significantly different from wild-type expression. The t-tests rely on a pooled estimate of the standard deviation of the measurement noise as well as estimates of the mean wild-type expression for each gene. To improve the power of the tests, one iterates between estimating the downstream effects of each knockout and updating the estimates of the means and standard deviation based until the influence matrix is left unchanged. Means are initialized to the wild-type observations and the standard deviation is initially based on all but the direct targets of each knockout. After each update of the influence matrix, the means and standard deviation were updated using the observations estimated to be unaffected by the knockouts.

Estimating the influence matrix using t-tests required specifying a nominal significance level, α. To do so, we plotted the estimated number of potential regulators for several values of α. We then chose α by looking for an 'elbow' where the slope of the curve sharply increases; see Figure C.2. After choosing α, additive ODEs were fit to the time-series and used to rank the potential edges. We set $\lambda_2 = 0$ due to the sparsity already introduced using the knockouts and chose λ_1 by GCV, searching $\lambda_1 \in \{10^z, z = -2, -1.5, -1, ..., .5, 1\}$ for the 100-node networks and $\lambda_1 \in \{10^z, z = -2, -1.8, -1.6, ..., .8, 1\}$ for the 10-node networks.

The results are in Table 4.7, and include comparisons to teams 315, 304, 256 from the competition for comparison. Again, the results compare favorably particularly considering we made no attempt to optimize the unranked edges eliminated by the prescreening step. We present these comparisons because team 315 was the top performer overall, while teams 304 and 256 were the top performers among those whose methods primarily made use of dynamic models. For this subset of teams, Team 304 was the top performer (fifth overall) on the 100-node networks and Team 256 was the best performer (third overall) on the 50-node networks. Team 304 included the developers of Inferelator, which was a primary component in their larger network reconstruction pipeline [50]. Notably, Team 256 also took a nonparametric approach and utilized ODEs albeit using Bayesian estimation and a different strategy for reconstructing the network from the fitted model [1]. Despite these similarities, our approach offers the advantage of being scalable.

Network	Method	AUC PR	AUC ROC
Ecoli 1	Additive ODE	0.16 (0.154, 0.163)	0.53 (0.519, 0.532)
Ecoli 1	Linear ODE	**0.20** (0.189, 0.200)	**0.60** (0.594, 0.608)
Ecoli 1	Linear ODE + Lasso	0.15 (0.150, 0.159)	0.46 (0.449, 0.461)
Ecoli 1	Inferelator 1.0	0.15 (0.146, 0.154)	0.49 (0.480, 0.494)
Ecoli 2	Additive ODE	0.20 (0.197, 0.204)	0.54 (0.537, 0.549)
Ecoli 2	Linear ODE	**0.25** (0.238, 0.253)	**0.58** (0.569, 0.583)
Ecoli 2	Linear ODE + Lasso	0.23 (0.229, 0.238)	0.50 (0.498, 0.506)
Ecoli 2	Inferelator 1.0	0.21 (0.207, 0.213)	0.52 (0.511, 0.520)
Yeast 1	Additive ODE	0.10 (0.102, 0.106)	0.45 (0.445, 0.456)
Yeast 1	Linear ODE	0.11 (0.110, 0.115)	0.45 (0.442, 0.452)
Yeast 1	Linear ODE + Lasso	0.12 (0.114, 0.119)	0.44 (0.434, 0.446)
Yeast 1	Inferelator 1.0	**0.22** (0.211, 0.220)	**0.56** (0.554, 0.565)
Yeast 2	Additive ODE	0.31 (0.307, 0.314)	0.53 (0.526, 0.536)
Yeast 2	Linear ODE	**0.36** (0.358, 0.367)	**0.59** (0.583, 0.593)
Yeast 2	Linear ODE + Lasso	0.27 (0.270, 0.278)	0.40 (0.397, 0.405)
Yeast 2	Inferelator 1.0	0.33 (0.325, 0.330)	0.45 (0.446, 0.453)
Yeast 3	Additive ODE	0.23 (0.228, 0.234)	0.48 (0.470, 0.481)
Yeast 3	Linear ODE	**0.31** (0.308, 0.319)	**0.56** (0.558, 0.571)
Yeast 3	Linear ODE + Lasso	0.28 (0.271, 0.279)	0.47 (0.463, 0.473)
Yeast 3	Inferelator 1.0	0.29 (0.290, 0.300)	0.48 (0.472, 0.484)

Table 4.5: *AUC-PR and AUC-ROC for DREAM 3 10-node networks.* Performance comparisons are for a single dataset generated using GeneNetWeaver. The simulated data set contains 10 multifactorial perturbations with 21 observed time points on each. The trajectories were simulated using ODEs only. Gaussian noise with standard deviation .025 was added prior to normalization. Figures shown are means with 95% confidence intervals computed from 500 realizations of the measurement noise.

Network	Method	AUC PR	AUC ROC
Ecoli 1	Additive ODE	**.109** (.106, .113)	**.639** (.627, .650)
Ecoli 1	Linear ODE	.020 (.016, .023)	.540 (.533, .547)
Ecoli 1	Linear ODE + Lasso	.022 (.018, .026)	.547 (.538, .556)
Ecoli 1	Inferelator 1.0	.067 (.059, .074)	**.622** (.611, .634)
Ecoli 2	Additive ODE	.038 (.036, .040)	**.658** (.646, .670)
Ecoli 2	Linear ODE	.021 (.014, .027)	.525 (.516, .534)
Ecoli 2	Linear ODE + Lasso	.020 (.017, .023)	.533 (.523, .543)
Ecoli 2	Inferelator 1.0	**.060**(.051, .069)	.599 (.589, .609)
Yeast 1	Additive ODE	.085 (.084, .087)	**.615** (.611, .620)
Yeast 1	Linear ODE	.053 (.047, .059)	**.609** (.601, .617)
Yeast 1	Linear ODE + Lasso	.045 (.040, .051)	.536 (.523, .549)
Yeast 1	Inferelator 1.0	**.100** (.094, .105)	.582 (.57, .594)
Yeast 2	Additive ODE	**.072** (.070, .074)	**.572** (.565, .579)
Yeast 2	Linear ODE	.048 (.048, .049)	**.568** (.563, .573)
Yeast 2	Linear ODE + Lasso	.045 (.044, .046)	.518 (.510, .526)
Yeast 2	Inferelator 1.0	.066 (.064, .067)	.517 (.513, .520)
Yeast 3	Additive ODE	.109 (.106, .111)	**.613** (.608, .619)
Yeast 3	Linear ODE	.094 (.092, .097)	**.611** (.605, .618)
Yeast 3	Linear ODE + Lasso	.089 (.087, .092)	.575 (.564, .587)
Yeast 3	Inferelator 1.0	**.118** (.115, .122)	.579 (.572, .587)

Table 4.6: *AUC-PR and AUC-ROC for DREAM3 100-node networks.* Performance comparisons are for a single dataset generated using GeneNetWeaver. The simulated data set contains 100 multifactorial perturbations with 21 observed time points on each. The trajectories were simulated using ODEs only. Gaussian noise with standard deviation .025 was added prior to normalization. The top performer(s) in each column are bolded. Figures shown are means with 95% confidence intervals computed from 10 realizations of the measurement noise.

	E1		E2		Y1		Y2		Y3	
	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC
Team 256, 10-Node	.396	.720	.258	.622	.258	.591	.481	.591	.434	.625
Team 304, 10-Node	.193	.697	.377	.791	.468	.909	.332	.554	.388	.658
Team 315, 10-Node	.710	.928	.713	.912	.897	.949	.541	.747	.627	.714
Additive ODEs, 10-Node	.875	.976	.632	.885	.558	.906	.491	.673	.510	.654
Team 304, 100-Node	.132	.835	.154	.879	.159	.839	.179	.738	.161	.667
Team 315, 100-Node	.694	.948	.806	.960	.493	.915	.469	.856	.433	.783
Additive ODEs, 100-Node	.623	.867	.841	.953	.466	.820	.424	.787	.396	.734

Table 4.7: *Results on DREAM 3 competition data* Performance on the *in silico* data for challenge 4 of the DREAM 3 competition [70, 52, 53]. Team 315 was the top performer in the challenge [93], while teams 304 and 256 focused on time series and were among the top performers. Team 304 used Inferelator 1.0 as part of a larger reconstruction pipeline [50]. Team 256 also fit non-parametric ODEs though their approach differed from ours [1]; team 256 did not participate in the 100-node reconstructions due to the complexity of their method. The competition data includes observations for wild-type, knockouts of each gene, heterozygous knockdowns of each gene, and time series under multifactorial perturbations. There are 21 observations on each of the 4 time series for the 10-node networks and the 46 time series for the 100-node networks. Our reconstructions used knockout and time-series data only. PR and ROC values were computed using GeneNetWeaver.

4.4 Discussion

This chapter introduces a novel technique, NeRDS, for reconstructing networks from time-series data. Unlike other ODE-based approaches which assume a parametric model, we take a nonparametric approach utilizing additive rather than linear approximations. We also introduce a coupling metric that can be used as a general tool for measuring the direct influence of one component on another in nonlinear ODE models. The flexibility of the nonparametric approach allows researchers to proceed with minimal assumptions other than the underlying smoothness inherent to ODE models.

While our approach is flexible, like any nonlinear approach it comes at the price of large data requirements. Specifically, for NeRDS to perform well we require as many time series as network components and that these time series be sufficiently informative. At a minimum, the trajectories of each component must exhibit enough curvature for its regulatory effects to be disambiguated from others on at least some of the time-series experiments. However, in general the number and quality of the time series is much more important than the frequency at which these time series our sampled, provided the sampling is sufficient to capture the system dynamics and maintain some signal amidst the noise. Moreover, time-series data tend to be more readily available then the more informative direct perturbation experiments, such as gene deletion.

Indeed, network reconstruction methods for time series currently lag techniques based on direct perturbations experiments. However their ability to make use of more readily available data is a major advantage, particularly in the early stages of understanding a system—precisely when network reconstruction is most relevant. Given the limitations of current time-series approaches, our method adds to the toolkit for network reconstruction and system identification. No single reconstruction method will be best in all cases. In fact, community network reconstructions that combine information from a variety of algorithms are often superior [54]. Further it expands the class of models available for time-course data to include additive ODEs, thus enriching the collection of methods available for community-based reconstructions.

The flexibility of our method must be balanced against both model and computational complexity. Central to managing these tradeoffs are the tuning parameters: λ_1 for controlling the smoothness of the additive functions and λ_2 for managing network-level sparsity. Larger choices for these parameters lead to simpler models, smaller choices to additional complexity. For instance, as $\lambda_1 \to \infty$ our additive model subsumes a linear models as a special case. The model complexity can also be reduced by limiting the number of interior knots in the basis expansions for the additive functions. While the GCV criterion offers an option for automatic tuning, it tends to err on the side of complexity. Diagnostic plots such as described in the supplement are an invaluable tool in making these selections subjectively. In practice, especially in an exploratory context, we recommend researchers start near the linear case and add additional complexity by decreasing λ_1 or adding knots as needed. Indeed, early simulation studies on the systems studied in this chapter demonstrated that allowing too much complexity (using too many knots or allowing λ_1 to be too small) significantly reduced performance of the additive ODEs.

Despite the relative complexity of NeRDS we were able to scale to the 100-node networks because the methodology is both modular and easily parallelized. A key reason for the latter is the marginal nature of the reconstruction method. Regression-based approaches such as the current one construct the network by combining the incoming edges selected (ranked) for each node individually. For a network with d nodes this allows the model fitting to be split into d separate tasks. Likewise, the most computationally intensive portion of our methodology—selecting tuning parameters—is trivially parallelized by splitting along each value of the tuning parameter considered in the grid search.

Moreover, by employing basis expansions the nonparametric method allowed us to expand the model class while still only needing to solve linear systems. In addition, since the submodels for all d nodes share a single feature space we need to compute only once the matrices defining these linear systems and the decompositions needed to efficiently solve them. Thus while tens of thousands of linear systems were solved in fitting our additive nonparametric models, only a few hundred matrix decompositions were required ($d = 100$ for each value of the smoothing parameter λ_1 considered).

In relatively small systems, such as the Mouse Embryonic Stem Cell and *Lactocaccus Lactis* systems serving as our primary examples, it appears preferable to fix the sparsity parameter λ_2 at zero in advance. In contrast, the role of sparsity becomes increasingly important as the number of network nodes grows into the tens and beyond. Moreover, inducing sparsity through λ_2 offers the potential to skirt the requirement of as many time series as nodes but at the expense of discovering fewer true edges.

Many of the tradeoffs discussed above are inherent in the problem of reconstructing biological networks and are by no means unique to our method. Generally, there is a continued need for theory to better understand the tradeoffs and how best to manage them. Theory is needed not just for managing tradeoffs within a modeling paradigm, but also for experimental design. Network reconstruction methods based on time series offer two advantages in this regard. First, they rely on the easiest to obtain data and so offering early insight on how to proceed with future experiments. Also, time series methods yield dynamic models useful for estimating the likely information gain from potential experiments.

In order to move toward genome-scale network reconstruction, further work will also be needed to explore how the method presented here fits in with efforts toward data integration. For instance, within the additive framework it is not obvious how to combine multiple time-series datasets not emanating from a single-lab or experimental setup as has been done for linear systems [90]. Determining how to integrate sources of data other than time series, including prior information, network motifs from homologous systems, and steady-state data from perturbation experiments, among others, is a promising direction for further research [40, 91, 33].

APPENDICES

APPENDIX A

Algorithms for Chapter II

Overview

This appendix collects algorithms used for carrying out structure MCMC with partial-order proposals over the space of DAGs. The main algorithm, Algorithm 2, appears with a detailed discussion of the partial order proposals in section A. For expositional clarity key portions of this algorithm are given separately as Algorithm 3 for proposing a partial order by removing a relation and Algorithm 4 for sampling relations formerly mediated by the one removed.

Section A presents Algorithm 5 for sampling partial orders from a linear order L. This algorithm is used by the partial order proposals but may also be of independent interest for attempting to do importance sampling over the space of graphs. A key property of this algorithm is that it provides a 1-1 mapping between the sequence of randomizations and the resulting partial order. Moreover, this mapping can be set up so that the probability of obtaining a given partial order is the same for all of its linear extensions. A formal statement and proof of this property is given in Proposition 1.1. In section A, we present Algorithm 6 which is used for sampling a DAG once a partial order has been proposed.

Recall the following notation: we are sampling the space of DAGs on a node set V with $|V| = d$, $s = \binom{d}{2}$ is the total number of pairs, and $\xi : V^2 \rightarrow \{1, \ldots, s\}$ is an indexing of node pairs.

Structure MCMC with Partial-Order Proposals

ALGORITHM 2 performs structure MCMC over the space of DAGs using Metropolis-Hastings. However, in contrast to proposals based on moving to adjacent graphs trough addition, removal, or reversal of edges, here DAGs are sampled uniformly from a partial order. At each step the algorithm makes a randomized decision on whether to sample the current partial order or a new one. The probability of doing so depends on the size of the partial order, i.e. how many graphs it contains relative to the maximum, and a control parameter ρ that determines the maximal probability of sampling the current partial order.

The heart of the algorithm is the partial-order proposal distribution $q_1(\pi'|\pi)$ presented at a high-level in ALGORITHM 3. This algorithm can best be understood as operating on the associated transitive reductions. Viewing the transitive reduction, $\breve{\pi}$, as a DAG with no feedforward motifs, the moves proposed consists of familiar edge additions and deletions. Edge additions are only allowed between node pairs (i, j) such that $i \,||_{\pi}\, j$. When forming $\breve{\pi}'$ by adding (i, j) to $\breve{\pi}$, we also add it to π' and remove from $\breve{\pi}'$ all edges that have become feedforward motifs after the addition. However, so that addition of (i, j) will be reversible by deletion of (i, j), there must be an opportunity for the removed edges to be restored as well. This is accomplished by randomized decisions to include $(h, k) \in \breve{\pi}'$ if removal of (i, j) severs the path in $\breve{\pi}$ from h to k. In other words, the algorithm samples to include (h, k) for all $h \in \bar{A}_i$, i and its ancestors, and $k \in \bar{D}_j$, j and its descendants. Of course, (i, j) is included without sampling. Not all of these 'severed'-relations are sampled for inclusion; instead they are sampled in a specific order to ensure there is a unique sequence of randomizations leading from $\breve{\pi}$ to $\breve{\pi}'$ conditional on the removal of (i,j). The edges that are sampled are done so with a fixed probability $p \in (0,1)$. Formally, define the paths *potentially* severed by removing i,j as the set of pairs,

$$S_{i||j} = \{(h,k) \neq (i,j) : h \in \bar{A}_i, k \in \bar{D}_j, |\pi_{\xi(h,k)}| > 0\}.$$

The modifier *potentially* is necessary here since not all paths connecting h to k, for $(h,k) \in S_{i||j}$, must traverse the edge (i,j). Relations in S can be sampled using Algorithm 4 which makes use of Algorithm 5 to ensure there is a unique sequence of randomizations leading to the proposal π'. The structure of the algorithm is such that (h,k) will always be included without sampling when there exists a path in $\breve{\pi}$ from h to k that does not involve (i,j).

Algorithm 2 Structure MCMC with Partial-Order Proposals

```
 1: **Input:** Initial structures $\gamma^0, \pi^0, \breve{\pi}^0$, control parameters $\rho$ and $p$, sample size $N$
 2: Set $m = (d-1)(d-2)/2$
 3: **for** n=1:N **do**
 4:     Sample $U_1, U_2 \sim U(0,1)$
 5:     **if** $U_1 < \rho 2^{||\pi^{n-1} - \breve{\pi}^{n-1}||_0 - m}$ **then**
 6:         $\gamma' \sim q_2(\cdot|\pi)$
 7:         $\alpha \leftarrow [p(\mathcal{D}|\gamma')p(\gamma')]/[p(\mathcal{D}|\gamma)p(\gamma)]$
 8:     **else**
 9:         $\pi' \sim q_1(\cdot|\pi;p)$
10:         $\gamma' \sim q_2(\cdot|\pi')$
11:         $\alpha \leftarrow [p(\mathcal{D}|\gamma')p(\gamma')q_2(\gamma|\pi)q_1(\pi|\pi';p)]/[p(\mathcal{D}|\gamma)p(\gamma)q_2(\gamma'|\pi')q_1(\pi'|\pi;p)]$
12:     **end if**
13:     **if** $U_2 < \alpha$ **then**
14:         $\gamma^n \leftarrow \gamma'$
15:     **else**
16:         $\gamma^n \leftarrow \gamma^{n-1}$
17:     **end if**
18:     $\pi^n \leftarrow \pi(\gamma^n)$
19:     $\breve{\pi}^n \leftarrow \breve{\pi}(\pi^n)$
20: **end for**
21: **Output:** A sample $\{\gamma_1, ..., \gamma_N\}$ from $p(\gamma|\mathcal{D})$
```

Algorithm 3 Partial Order Proposals

```
1: Input: partial order $\pi$, transitive reduction $\breve{\pi}$, and control parameter $p \in (0, 1)$
2: Initialize $\pi' \leftarrow \pi$
3: Select a node $i$ at random
4: Find $J_0(i) = \{j : \pi_{\xi(i,j)} = 0\}$ and $J_1(i) = \{j : j \in \mathrm{ch}_i(\breve{\pi})\}$.
5: if $J_0(i) \cup J_1(i) = \emptyset$ then
6:     return to beginning and pick a new node $i$
7: else
8:     Select $j$ from $J_0(i) \cup J_1(i)$ at random
9: end if
10: if $j \in J_0(i)$ then
11:     Set $\pi'_{\xi(i,j)} \leftarrow 1[i < j] - 1[i > j]$ and form $\breve{\pi}'$
12: else
13:     Set $\pi'_{\xi(i,j)} \leftarrow 0$ and $\breve{\pi}'_{\xi(i,j)} \leftarrow 0$
14:     Update $\pi'$ and $\breve{\pi}'$ by systematically sampling severed paths $S_{i||j}$.
15: end if
16: Output: partial order proposal $\pi'$ and $\breve{\pi}'$
```

Algorithm 4 removing (i, j) from $\breve{\pi}$ and sampling severed paths

```
1: Input: $\pi$, $\breve{\pi}$, $L$, $(i, j)$ and $p$
2: Set $\mathrm{p}_{hk} \leftarrow 1[(h, k) \in \breve{\Pi}(\pi)]$ for all pairs $(h, k)$
3: Set $\mathrm{p}_{hk} \leftarrow .5$ for all $(h, k) \in S_{i||j}$
4: Set $\mathrm{p}_{ij} \leftarrow 0$
5: Sample $\pi'$ using ALGORITHM 5 with preferences p
6: Output: $\pi', \breve{\pi}'$
```

Proofs for Lemmas 2.1 and 2.2

Lemma 2.1 *The Markov Chains generated by* Algorithm *2 are reversible and have stationary distribution $p(\gamma|\mathcal{D})$.*

Proof: Following Hastings [36], we will show that transition probabilities and $p(\gamma|\mathcal{D})$ satisfy detailed balance implying the statements above.

For $\gamma \neq \gamma'$ the transition probabilities are given by the product of the probabilities for proposing a move and accepting the move,

$$\kappa(\gamma'|\gamma) = q_2(\gamma'|\breve{\pi}')q_1(\breve{\pi}'|\breve{\pi}) \times \alpha(\gamma'|\gamma), \tag{A.1}$$

where the acceptance ratio is,

$$\alpha(\gamma'|\gamma) = \frac{p(\gamma'|\mathcal{D})q_2(\gamma|\breve{\pi})q_1(\breve{\pi}|\breve{\pi}')}{p(\gamma|\mathcal{D})q_2(\gamma'|\breve{\pi}')q_1(\breve{\pi}'|\breve{\pi})} \wedge 1. \tag{A.2}$$

By the complement rule,

$$\kappa(\gamma|\gamma) = 1 - \sum_{\gamma'\neq\gamma} q(\gamma'|\gamma)\alpha(\gamma'|\gamma) \tag{A.3}$$

which includes the probability that the proposal is in fact γ. Hence,

$$\begin{aligned} p(\gamma'|\mathcal{D})\kappa(\gamma|\gamma') &= p(\gamma'|\mathcal{D}) \times q_2(\gamma|\breve{\pi})q_1(\breve{\pi}|\breve{\pi}') \times \frac{p(\gamma|\mathcal{D})q_2(\gamma'|\breve{\pi}')q_1(\breve{\pi}'|\breve{\pi})}{p(\gamma'|\mathcal{D})q_2(\gamma|\breve{\pi})q_1(\breve{\pi}|\breve{\pi}')} \\ &= p(\gamma|\mathcal{D})q_2(\gamma'|\breve{\pi}')q_1(\breve{\pi}'|\breve{\pi}) \wedge p(\gamma'|\mathcal{D})q_2(\gamma|\breve{\pi})q_1(\breve{\pi}|\breve{\pi}'), \end{aligned} \tag{A.4}$$

and by the symmetric argument,

$$p(\gamma|\mathcal{D})\kappa(\gamma'|\gamma) = p(\gamma'|\mathcal{D})q_2(\gamma|\breve{\pi})q_1(\breve{\pi}|\breve{\pi}') \wedge p(\gamma|\mathcal{D})q_2(\gamma'|\breve{\pi}')q_1(\breve{\pi}'|\breve{\pi}). \tag{A.5}$$

Putting the two together we have the detailed balance condition,

$$p(\gamma'|\mathcal{D})\kappa(\gamma|\gamma') = p(\gamma|\mathcal{D})\kappa(\gamma'|\gamma). \tag{A.6}$$

□

Lemma 2.2 *The Markov Chains generated by* Algorithm *2 are irreducible.*

Proof: Let $\gamma, \gamma' \in \Gamma$ be two distinct DAGs. It is required to show that the hitting time for γ' is finite with positive probability conditional on the chain having started at γ; i.e. $P_\gamma(\sigma_{\gamma'} < \infty) > 0$ where $\sigma_\gamma = \inf\{n \geq 0 : \gamma_n = \gamma\}$. This can be shown by constructing an explicit sequence of moves leading from γ to γ'.

Let $\breve{\pi}$ and $\breve{\pi}'$ be the associated transitive reductions (viewed as sets) and $\xi : V \times V \to \binom{d}{2}$ an ordering on the nodes. Let $R_1, \dots, R_{|\breve{\pi}|} \in \breve{\pi}$ such that $\xi(R_1) < \cdots < \xi(R_n)$ order the relations in $\breve{\pi}$. Define $\breve{\pi}_0 = \breve{\pi}$ and $\breve{\pi}_t = \breve{\pi}_{t-1} \setminus R_t$ and observe that $\breve{\pi}_{|\breve{\pi}|} = \emptyset$. Likewise, let $R'_1, \dots, R'_{|\breve{\pi}'|} \in \breve{\pi}'$ such that $\xi(R'_1) < \cdots < \xi(R'_n)$ order the relation in $\breve{\pi}'$ and define $\breve{\pi}'_0 = \emptyset$ and $\breve{\pi}'_t = \breve{\pi}_{t-1} \cup R_t$ so that $\breve{\pi}'_{|\breve{\pi}'|} = \breve{\pi}'$. Finally, choose arbitrary DAGs consistent with each partial order defined above, $g_t \in \Gamma_{\breve{\pi}_t}$ and $g'_s \in \Gamma_{\breve{\pi}'_s}$ for $t = 1, \dots, |\breve{\pi}|$ and $s = 1, \dots, |\breve{\pi}'|$. We have,

$$\begin{aligned}
P_\gamma(\sigma_{\gamma'} < \infty) &\geq P_\gamma(\sigma_\emptyset < \infty)P_\emptyset(\sigma_{\gamma'} < \infty) \\
&> \prod_{t=1}^{|\breve{\pi}|} \frac{p(g_t|\mathcal{D})q_2(g_{t-1}|\breve{\pi}_{t-1})q_1(\breve{\pi}_{t-1}|\breve{\pi}_t)}{p(g_{t-1}|\mathcal{D})q_2(g_t|\breve{\pi}_{t-1})q_1(\breve{\pi}_t|\breve{\pi}_{t-1})} \times \\
&\quad \prod_{t=1}^{|\breve{\pi}'|} \frac{p(g'_t|\mathcal{D})q_2(g'_{t-1}|\breve{\pi}'_{t-1})q_1(\breve{\pi}'_{t-1}|\breve{\pi}'_t)}{p(g'_{t-1}|\mathcal{D})q_2(g'_t|\breve{\pi}'_{t-1})q_1(\breve{\pi}'_t|\breve{\pi}'_{t-1})} \\
&> 0
\end{aligned} \tag{A.7}$$

where the last inequality follows because each term in the product is positive. □

Backward Partial Order Sampler

ALGORITHM 5 is a backward sampler for sampling a partial order π consistent with a linear order, L. Aside from L, it takes as input a matrix of pairwise preferences, $\mathrm{p} \in [0,1]^{d\times d}$, used to determine the probability a particular relation is included in the partial order when this relation is *sampled* rather than *required*. We call it a *backward* sampler because the outermost loop, indexed by i, begins with the last but one element in the linear order and then proceeds backwards through the linear order. In the inner loop, indexed by j, relations from L_i to L_j are added according to the following scheme. Whenever a relation is not *required*, it is included with probability determined by the appropriate element in p; initially, no relations are required. Each time a relation is added, we *require* all additional relations needed to maintain transitivity. When a relation is added because it is required rather than sampled, it does not contribute to the probability of drawing that particular partial order. This is important because it means there is a unique sequence of randomizations leading from L to π.

Recalling that $\check{\Pi}$ is the transitve reduction of Π, where Π is the partial order π viewed as a set, the *required* relations are exactly those in $\Pi \setminus \check{\Pi}$. Using this notation, the (log)-probability of sampling Π from L using ALGORITHM 5 is,

$$\begin{aligned}
&\log q_1(\Pi; L, \mathrm{p}) \\
&\quad = \textstyle\sum_{1\le i<j\le d} 1[(i,j)\in \Pi^c \uplus \check{\Pi}] \Big\{ 1[(i,j)\in\Pi] \log \mathrm{p}_{L_iL_j} + \\
&\qquad\qquad 1[(i,j)\in\Pi^c] \log(1-\mathrm{p}_{L_iL_j}) \Big\} \\
&\quad = \textstyle\sum_{(i,j)\in\check{\Pi}} \log \mathrm{p}_{L_iL_j} + \sum_{(i,j)\in\Pi^c} \log(1-\mathrm{p}_{L_iL_j}). \qquad \text{(A.8)}
\end{aligned}$$

Proposition 1.1. *Let L and L' be distinct linear extensions of a partial order π. Under* ALGORITHM *5, $q_1(\pi; L, p) = q_1(\pi; L', p)$ provided for all $(i,j) \notin \Pi(\pi)$, $p_{ij} = p_{ji}$.*

Proof: First observe that $\Pi \subset L$ for all $L \in \mathcal{L}_\pi$. Hence, for all $(i,j) \in \Pi$ we have $i <_L j$ for all linear extensions L. If $(i,j) \in \check{\Pi}$ then ALGORITHM 5 includes (i,j) with probability p_{ij} for all such L. If $(i,j) \in \Pi/\check{\Pi}$ the edge is included without sampling under all linear extensions, since for any mediating path $i \prec_\pi m_1 \prec_\pi ... \prec_\pi m_\ell \prec_\pi j$ we have also $i <_L m_1 <_L ... <_L m_\ell <_L j$ for all $L \in \mathcal{L}_\pi$. Thus the first sum in (A.8) is equal for all $L \in \mathcal{L}_\pi$. It follows directly from the assumption that the second sum is equal for all linear extensions. □

Remark 1.2. By the proposition above, if it is desired that all sampled π be equiprobable under all linear extensions, the preferences p must either equally prefer $i \prec_\pi j$ and $j \prec_\pi i$ relative to $i||_\pi j$ or take one of $\mathrm{p}_{ij}, \mathrm{p}_{ji}$ to be one and the other zero. This is always the case when called by ALGORITHM 2.

Algorithm 5 Backward π-sampler: sample a partial order π from a linear order L with preferences p

```
1: Input: linear order $L \in \sigma(V)$, preferences $\mathrm{p} \in [0,1]^{d\times d}$
2: Initialize: $\pi = \check{\pi} = (0,\dots,0) \in \mathbb{Z}^s$, $r = (0,...,0) \in \mathbb{N}^s$, $q = 0$
3: for $i = (d-1) : 1$ do
4:     for $j = (i+1) : d$ do
5:         if $r_{\xi(L_i,L_j)} = 1$ then $\pi_{\xi(L_i,L_j)} \leftarrow 1[L_i < L_j] - 1[L_i > L_j]$
6:         else sample $u \sim$ Uniform(0,1)
7:             if $u < \mathrm{p}_{L_iL_j}$ then
8:                 $\check{\pi}_{\xi(L_i,L_j)}, \pi_{\xi(L_i,L_j)} \leftarrow 1[L_i < L_j] - 1[L_i > L_j]$
9:                 and $q \leftarrow q + \log(p_{L_iL_j})$
10:            else $q \leftarrow q + \log(1 - p_{L_iL_j})$
11:            end if
12:        end if
13:    end for
14:    if $j < d$ then
15:        for $k = (j+1) : d$ do
16:            if $\pi_{\xi(L_j,L_k)} \neq 0$ then
17:                $r_{\xi(L_i,L_k)} \leftarrow 1$
18:            end if
19:        end for
20:    end if
21: end for
22: Output: $\pi \in \{-1,0,1\}^s$, transitive reduction $\check{\pi}$, and log probability $q$
```

DAG-Sampler

Algorithm 6 is used to sample DAGs specified as shortest paths between pairs in V. It takes as input a partial order π, any linear extension L, and a matrix of probabilities D. Here D_{ij} is the probability with which edge $i \to j$ is included when sampled. Typically, $D \equiv p_\gamma$ for $p_\gamma \in (0, 1)$. However, unequal preferences are equally valid and may be useful for improving acceptance rates in some settings. The algorithm returns a DAG consistent with π by independently sampling feedforward edges as they are encountered. Observe that for $p_\gamma = .5$ the algorithm uniformly selects a DAG from Γ_π.

Algorithm 6 γ-sampler: sample a DAG from a partial order

```
1: **Input:** partial order $\pi$, linear extension $L$, probability matrix $D \in [0,1]^{d\times d}$
2: **Initialize:** $\gamma = (0,\dots,0) \in \mathbb{Z}^s$, $q_2 = 0$
3: **for** $i = (d-1) : 1$ **do**
4:     **for** $j = (i+1) : d$ **do** $z \leftarrow \xi(L_i, L_j)$
5:         **if** $\pi_z \neq 0$ **then**
6:             **if** $j = i+1$ **then** $\gamma_z \leftarrow \pi_z$
7:             **else** $\ell \leftarrow d$
8:                 **for** $k = (i+1) : (j-1)$ **do**
9:                     **if** $|\gamma_{\xi(L_i,L_k)}| = 1$ and $\gamma_{\xi(L_k,L_j)} \neq 0$ **then** $\ell \leftarrow |\gamma_{\xi(L_j,L_k)}| + 1 \wedge \ell$
10:                    **end if**
11:                **end for**
12:                **if** $\ell < d$ **then** sample $u \sim$ Uniform(0,1)
13:                    **if** $u < D_{z,\ell}$ **then** $\gamma_z \leftarrow \ell\pi_z$ and $q_2 \leftarrow q_2 + \log(D_{z,\ell})$
14:                    **else** $\gamma_z \leftarrow \pi_z$ and $q_2 \leftarrow q_2 + \log(1 - D_{z,\ell})$
15:                    **end if**
16:                **else** $\gamma_z \leftarrow \pi_z$
17:                **end if**
18:            **end if**
19:        **end if**
20:    **end for**
21: **end for**
22: **Output:** DAG $\gamma \in \{-(d-1),\dots,(d-1)\}^s$ and log probability $q_2$
```

APPENDIX B

Algorithms for Chapter IV

Sparse Backfitting

Algorithm 7 Sparse Backfitting

```
1: for $i = 1 : d$ do
2:     input Concatenated derivative estimates: $\hat{\dot{\boldsymbol{x}}}_i = (\hat{\dot{x}}_i^1(t_1), ..., \hat{\dot{x}}_i^R(t_n))'$.
3:     input Concatenated trajectory estimates: $\hat{\boldsymbol{x}}_j = (\hat{x}_j^1(t_1), ..., \hat{x}_j^R(t_n))'$,
4:             $j = 1, ..., d$.
5:     input Smoothing parameter $\lambda_{1i}$ and sparsity parameter $\lambda_{2i}$.
6:     initialize $\boldsymbol{f}_{ij} \leftarrow 0 \in \mathbb{R}^N$, for $j = 1, ..., d$, where $N = nR$.
7:     compute $\hat{\alpha}_i \leftarrow N^{-1} \sum_{k=1}^{N} \hat{\dot{\boldsymbol{x}}}_{ik}$.
8:     repeat
9:         Store starting values $\boldsymbol{f}_i^\star \leftarrow \boldsymbol{f}_i$.
10:        for $j = 1 : d$ do
11:            Compute residuals $\boldsymbol{r} \leftarrow \hat{\dot{x}}_i - \hat{\alpha}_i - \sum_{\ell \neq j} \boldsymbol{f}_{i\ell}(\hat{\boldsymbol{x}}_\ell)$
12:            Smooth $\boldsymbol{f}_{ij} \leftarrow S_j(\lambda_{1i})\boldsymbol{r}$
13:            Estimate Norm $s^2 \leftarrow \left(N^{-1} \sum_{k=1}^{N} \boldsymbol{f}_{ijk}^2\right)^{\frac{1}{2}}$
14:            Soft Threshold $\boldsymbol{f}_{ij} \leftarrow (1 - \lambda_{2i}/s^2)_+ \boldsymbol{f}_{ij}$
15:            Center $\boldsymbol{f}_{ij} \leftarrow \boldsymbol{f}_{ij} - \bar{\boldsymbol{f}}_{ij}$
16:        end for
17:    until Convergence $\sup_{j=1,...,d} \sup_{k=1,...,N} |\boldsymbol{f}_{ijk} - \boldsymbol{f}_{ijk}^\star| < \epsilon$.
18: end for
```

NeRDS Overview

Algorithm 8 NeRDS Workflow

```
1:  Stage 1 - Normalize and Smooth the Data
2:  for components $i = 1 : d$ do
3:      input time-course data $(Y_i^r(t_k))_{k=1,r=1}^{n,R}$
4:      compute $M_i = \max_{k,r}(Y_i^r(t_k))_{k=1,r=1}^{n,R}$
5:      compute $(\tilde{Y}_i^r(t_k))_{k=1,r=1}^{n,R} = (Y_i^r(t_k)/M_i)_{k=1,r=1}^{n,R}$
6:      for experiments $r = 1 : R$ do
7:          input normalized time-course data $(\tilde{Y}_i^r(t_k))_{k=1}^{n}$
8:          compute the trajectory $\hat{x}_i^r(t)$ using smoothing splines
9:          compute derivative $\hat{\dot{x}}_i^r(t) = \frac{d}{dt}\hat{x}_i^r(t)$
10:     end for
11: end for
12:
13: Stage 2 - Fit an Additive ODE
14: for components $i = 1 : d$ do
15:     input the derivatives $(\hat{\dot{x}}_i^r(t))_{r=1}^{R}$ as response
16:     input the trajectories $\{(\hat{x}_j^r(t))_{r=1}^{R}, j = 1, ..., d\}$ as features
17:     input parameters $\Lambda_1$ (smoothing) and $\Lambda_2$ (sparsity) to search
18:
19:     *Select Tuning Parameters*
20:     for $\lambda = (\lambda_1, \lambda_2) \in \Lambda_1 \times \Lambda_2$ do
21:         compute $\hat{f}_{i+}(\lambda)$ using Sparse Backfitting [ALGORITHM S2]
22:         compute $GCV(\lambda)$ [equation (11)]
23:     end for
24:     set $\lambda_i \leftarrow \arg\min_{\lambda \in \Lambda_\times \Lambda_2} GCV(\lambda)$
25:
26:     output $\hat{f}_{i+}(\lambda_i) = \sum_{j=1}^{d} \hat{f}_{ij}(x; \lambda_i)$
27: end for
28:
29: Stage 3 - Compute Coupling
30: for i=1:d do
31:     for j=1:d do
32:         input $\hat{f}_{ij}$
33:         input $\mathcal{R}_j = \text{range}(\hat{x}_j)$
34:         compute coupling $\hat{\rho}_{ij}(\hat{f}_{ij}) = \sqrt{\int_{\mathcal{R}_j} [\hat{f}_{ij}(x)]^2 dx / |\mathcal{R}_j|}$ [equation (16)]
35:     end for
36: end for
```

APPENDIX C

Diagnostics and Additional Results for Chapter IV

Diagnostics for tuning parameter selection

In order to fit an additive ODE to a collection of time series using NeRDS, four types of tuning parameters need to be selected. These are: (1) λ_0 controlling smoothness in the first stage; (2) λ_1 and the number of knots, controlling smoothness of the additive functions in stage 2; and (3) λ_2 controlling sparsity of the additive components in stage 2.

In the first stage, smoothing splines are fit to each component ($i = 1, ..., d$) of each time series ($r = 1, ..., R$) yielding estimates of the trajectories $\hat{x}_i^r(\cdot)$ and derivatives $\hat{\dot{x}}_i^r(\cdot)$. The smoothness of these trajectories are determined by smoothing parameters λ_{i0}^r and can be efficiently selected by GCV (which we use) or even standard cross validation. Once these estimates are obtained, it is useful to overlay plots of the observations $Y_i^r(t_k)$ and the estimated trajectories $\hat{x}_i^r(t)$ against time for each experiment r. See FIGURE C.1, panel A for an example. If the any of the estimated trajectories appear jagged, they may be over fit and λ_{i0}^r should be adjusted upwards to give a smoother estimate. This is of particular importance when the sampling density is low relative to the amount of noise.

For each submodel ($i = 1, ..., d$) in the second stage the smoothness of the additive components ($j = 1, ..., d$) is determined by λ_{i2} and the number of knots. As presented in the paper, this smoothing parameter can also be selected by GCV though this will often lead to overfitting due the noise in the derivatives $\hat{\dot{x}}_i^r(\cdot)$. As before, overlaying plots of the estimated derivatives and select additive fits versus time for each experiment r is useful for appropriately balancing flexibility and complexity.

As an example FIGURE C.1 panel B shows such plots for submodel 3 (Nanog) from the mouse system. Based on the smooths from the first stage (panel A), experiments $r = 4$ and $r = 5$ appear relatively unimportant due to the small rate of change in the trajectories of Nanog. For the remaning four experiments, $r = 1, 2, 3, 6$ we overlay plots of the estimated derivatives (black, solid), the linear fit (red, dashed), and an additive fit with 4 knots, $\lambda_2 = 0$, and $\lambda_1 = .01$ (cyan, dot-dash).

The additive model provides a better fit than the linear model while still being smoother than the estimated derivative it approximates indicating an appropriate balance between complexity and flexibility. If the fit is inadequate additional flexibility can be added by reducing λ_1 or incorporating additional knots. In contrast, if a similar fit can be achieved with larger λ_1 or fewer knots the simpler model should be preferred. In our simulations with the mouse system, we chose λ_1 by GCV but set the lowest value in the line search to .01 based on these plots. This served to prevent overfitting while allowing GCV to choose still less complexity when warranted by the data.

The final tuning parameter to be set is λ_2 controlling sparsity of the additive fits in stage 2. In settings where d is small, our experience in simulations has been that setting $\lambda_2 = 0$ or very small is best. However, for larger d, like in the DREAM 3 100-node networks, choosing $\lambda_2 > 0$ not only induce sparsity in the fitted submodels but also greatly speeds computation. When employing prescreening as we did with the DREAM 3 competition data, it may be possible to choose $\lambda_2 = 0$ as long as the number of potential regulators in each submodel is small. For λ_2 large enough, the null model with all the additive components identically zero will be returned. In practice, one can examine the same plots as for λ_1 to determine if a particular value of the sparsity parameter λ_2 allows adequate flexibility. One may wish to find a λ_2 leading to the null model and then progressively decrease until either: (1) adequate fit is seen in diagnostic plots; (2) the number of non-zero components is the additive fits are sufficiently large relative to prior knowledge on the approximate in-degree; (3) the computational burden becomes too much to decrease further. GCV remains an option for jointly determining λ_1 and λ_2 when looking at plots is impractical.

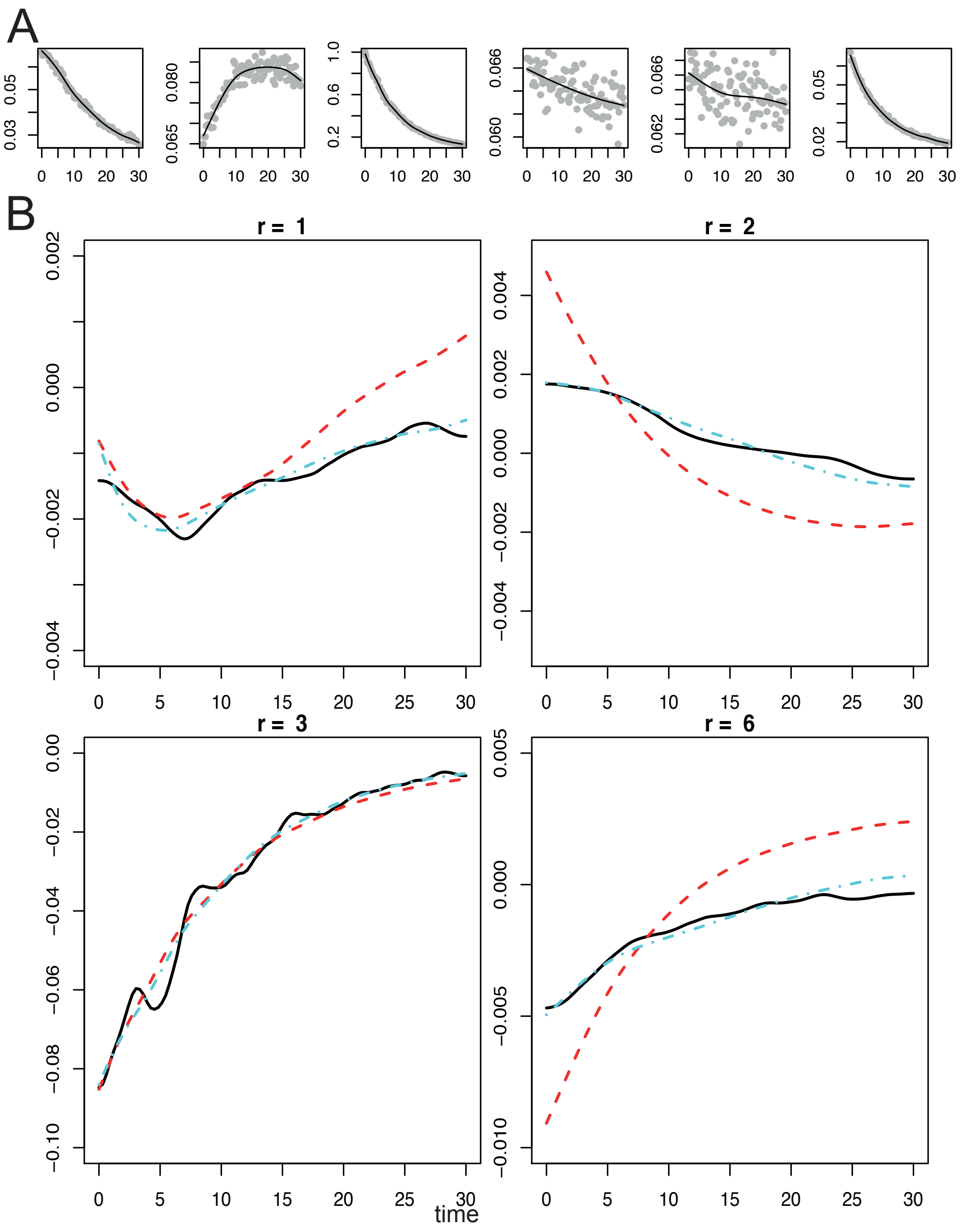


Figure C.1: *Diagnostic plots for component 3 (Nanog) in the mouse system.* **Panel A**: Each plot show the normalized observations from one the six simulated experiments as grey dots and the stage-1 smooth as a solid black line. Experiments 4 and 5 appear to carry minimal information for fitting a model to Nanog and are not considered in stage-2 diagnostics. **Panel B:** For each of four relevant experiments, the solid black line is the estimated derivative of Nanog, the dashed red line the unregularized linear fit, and the dot-dash cyan line the additive fit with $\lambda_1 = .01$, $\lambda_2 = 0$. The additive model provides a better fit on the non-dominant experiments.

Choosing α

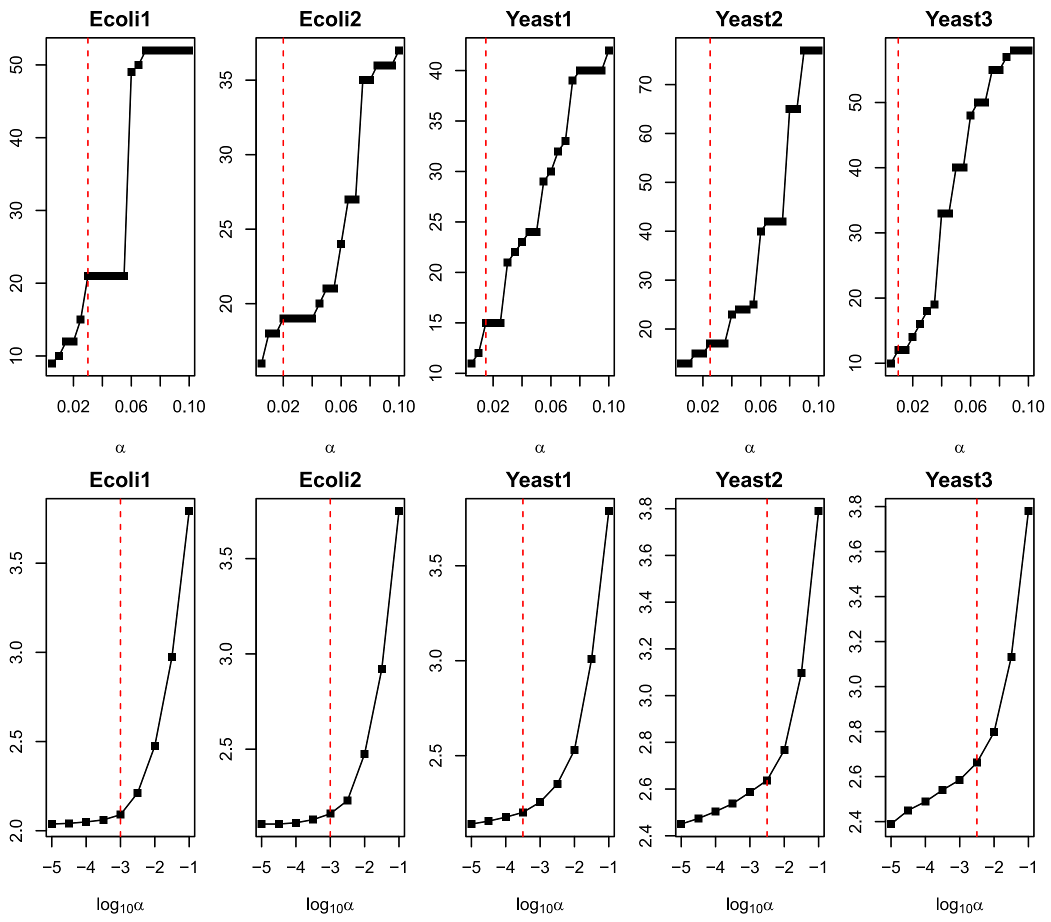


Figure C.2: *Plotting the number of potential edges versus α.* For the DREAM 3, challenge 4, competition data we used knockout experiments to limit the number of potential regulators. The algorithm used to do this relies on t-tests to determine which gene expression levels in each gene deletion mutant significantly differ from their wild-type expressions. The plots show the number of potential regulators versus the nominal significance level, α, used in these t-tests. We chose α by looking for an 'elbow'—a location where the slope of the curve sharply increases. The locations indicated by the dashed vertical lines were used for the results presented in Table 4.6; from left to right these are .03, .02, .015, .025, .015 for the 10-node networks and $10^z, z = -3, -3, -3.5, -2.5, -2.5$ for the 100-node networks. For the 100-node networks both the number of potential regulators and α are on the $\log_{10}$ scale. *Top Row:* 10-node networks. *Bottom row:* 100-node networks.

Varying the Sample Size

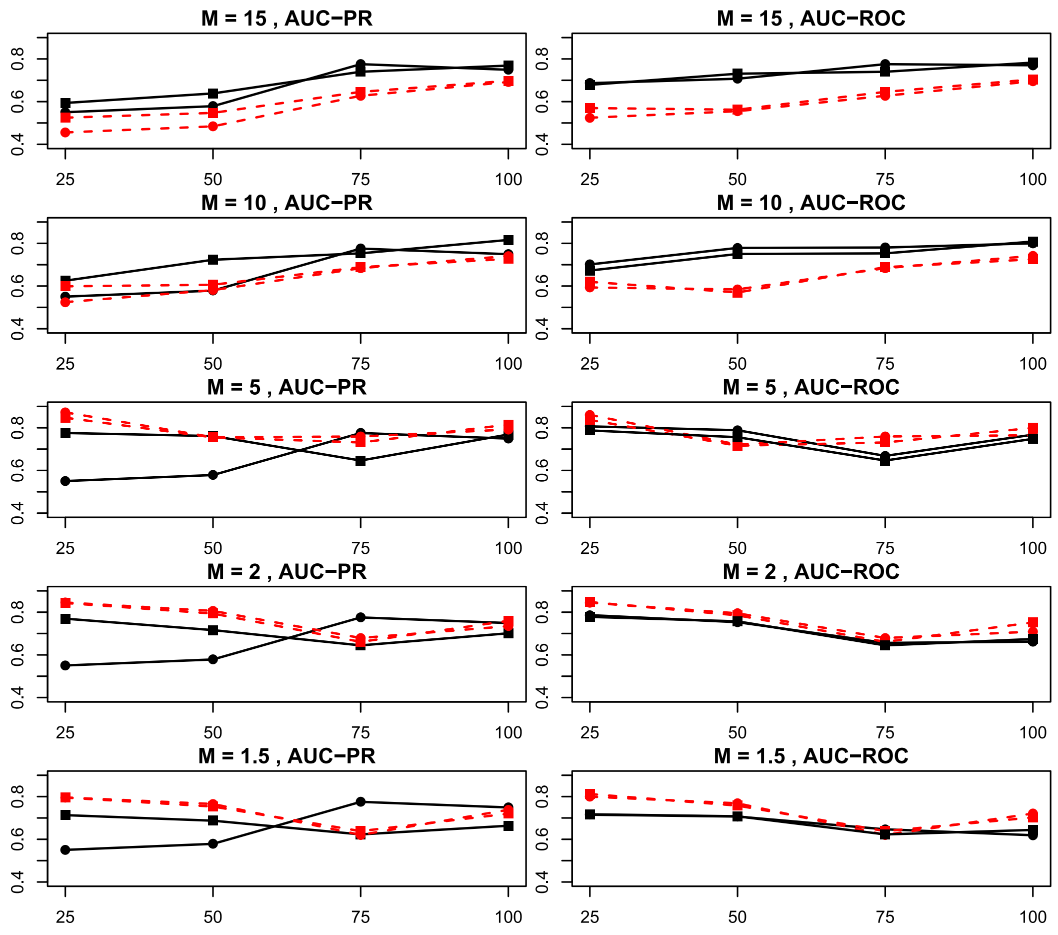


Figure C.3: *Reconstruction performance on the Lactococcus Lactis network for varied sample size.* Using the experimental setup for *Lactococcus* described in the Results section of the main paper, we repeated the simulations with reduced sampling densities. The number of observations per time series, n, is on the horizontal axis of each plot. Solid, black lines show the performance of the additive ODEs introduced in the paper while dashed, red lines indicate the performance for linear ODEs. The two noise levels, $\sigma \in \{.02, .05\}$, are respectively indicated by round and square symbols. For $n = 100$ these are the same results presented in Table 4.1 and Table 4.3.1.

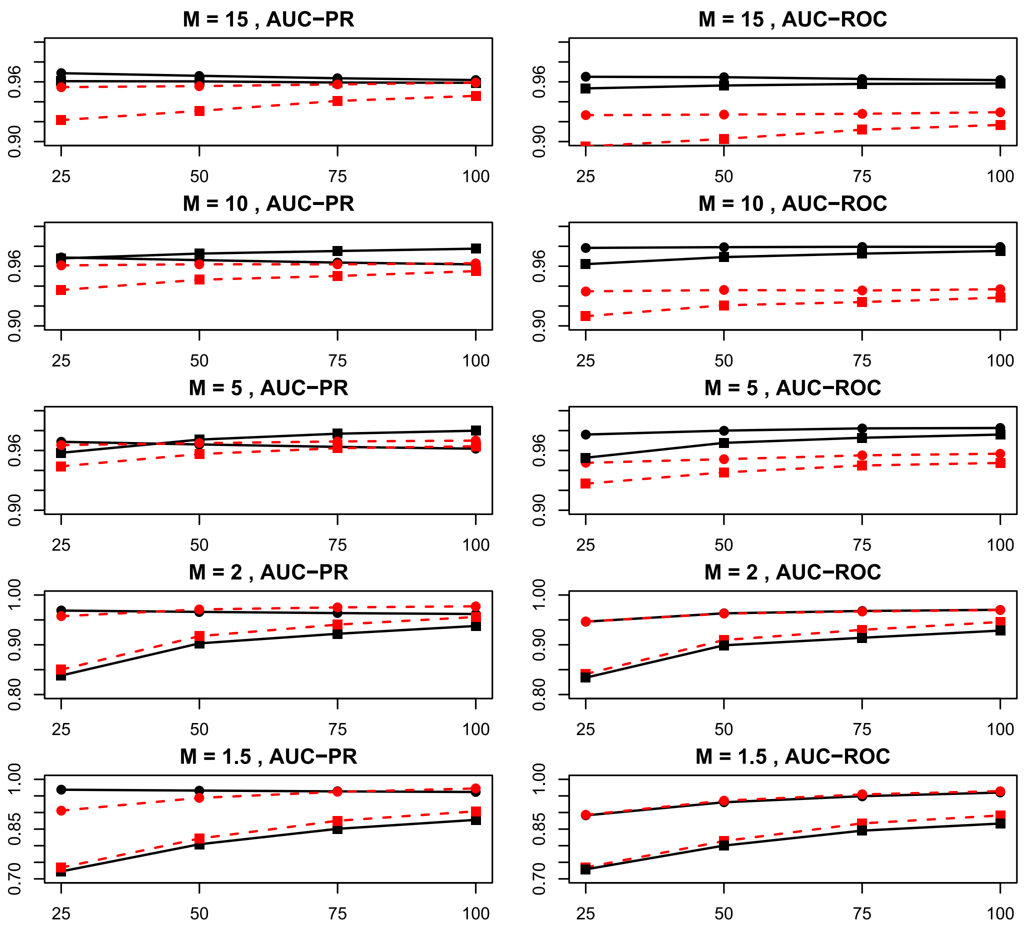


Figure C.4: *Reconstruction performance on the mouse network for varied sample size.* Using the experimental setup for the mouse network described in the Results section, we repeated the simulations with reduced sampling densities indexed by the number observations per time series, n, on the horizontal axes. Solid, black lines indicate the performance of the additive ODEs while dashed, red lines show the performance for linear ODEs. The two noise levels, $\sigma \in \{.02, .05\}$, are indicated by round and square symbols, respectively. For $n = 100$ these are the same results presented in TABLE 4.3.2 and TABLE 4.4.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Äijö, T., and H. Lähdesmäki (2009), Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics, *Bioinformatics*, *25*(22), 2937 –2944, doi:10.1093/bioinformatics/btp511.

[2] Andersson, S. A., D. Madigan, M. D. Perlman, et al. (1997), A characterization of markov equivalence classes for acyclic digraphs, *The Annals of Statistics*, *25*(2), 505–541.

[3] Banerjee, O., L. El Ghaoui, and A. d'Aspremont (2008), Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data, *The Journal of Machine Learning Research*, *9*, 485–516.

[4] Bard, Y. (1974), *Nonlinear parameter estimation*, Academic Press, New York.

[5] Bonneau, R., D. J. Reiss, P. Shannon, M. Facciotti, L. Hood, N. S. Baliga, and V. Thorsson (2006), The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo, *Genome Biology*, *7*(5), R36.

[6] Brightwell, G., and P. Winkler (1991), Counting linear extensions, *Order*, *8*(3), 225–242.

[7] Brunel, N. J.-B. (2008), Parameter estimation of ODEs via nonparametric estimators, *Electronic Journal of Statistics*, *2*(0), 1242–1267, doi:10.1214/07-EJS132.

[8] Buja, A., T. Hastie, and R. Tibshirani (1989), Linear smoothers and additive models, *The Annals of Statistics*, *17*(2), 453–510, ArticleType: research-article / Full publication date: Jun., 1989 / Copyright 1989 Institute of Mathematical Statistics.

[9] Buntine, W. (1991), Theory refinement on bayesian networks, in *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*, pp. 52–60, Morgan Kaufmann Publishers Inc.

[10] Cappé, O., E. Moulines, and T. Rydén (2005), Sequential monte carlo methods, *Inference in Hidden Markov Models*, pp. 209–250.

[11] Chickarmane, V., and C. Peterson (2008), A computational model for understanding stem cell, trophectoderm and endoderm lineage determination, *PLoS One*, *3*(10), e3478.

[12] Chickering, D. M. (2003), Optimal structure identification with greedy search, *The Journal of Machine Learning Research*, *3*, 507–554.

[13] Chou, I.-C., and E. O. Voit (2009), Recent developments in parameter estimation and structure identification of biochemical and genomic systems, *Mathematical Biosciences*, *219*(2), 57–83, doi:10.1016/j.mbs.2009.03.002.

[14] Cook, J. R., and L. A. Stefanski (1994), Simulation-extrapolation estimation in parametric measurement error models, *Journal of the American Statistical Association*, *89*(428), 1314–1328.

[15] Cooper, G. F., and C. Yoo (1999), Causal discovery from a mixture of experimental and observational data, in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 116–125, Morgan Kaufmann Publishers Inc.

[16] De Jong, H. (2002), Modeling and simulation of genetic regulatory systems: a literature review, *Journal of Computational Biology*, *9*(1), 67–103.

[17] Dempster, A. P. (1972), Covariance selection, *Biometrics*, pp. 157–175.

[18] Eaton, D., and K. Murphy (2007), Bayesian structure learning using dynamic programming and MCMC, *UAI*, pp. 101–108.

[19] Ellis, B., and E. Wong (2008), Learning causal bayesian network structures from experimental data, *JASA*, *103*, 778–789.

[20] Ellner, S. P., Y. Seifu, and R. H. Smith (2002), Fitting population dynamic models to time-series data by gradient matching, *Ecology*, *83*(8), 2256–2270.

[21] Foygel, R., and M. Drton (2010), Extended bayesian information criteria for gaussian graphical models, in *Advances in Neural Information Processing Systems*, pp. 604–612.

[22] Friedman, J., T. Hastie, and R. Tibshirani (2008), Sparse inverse covariance estimation with the graphical lasso, *Biostatistics*, *9*(3), 432–441.

[23] Friedman, N., and D. Koller (2003), Being bayesian about network structure: A bayesian approach to structure discovery in bayesian networks, *Machine Learning*, *50*, 95–126.

[24] Fujita, A. E., J. A. O. R. Sato, H. M. Garay-Malpartida, M. C. Sogayar, C. E. Ferreira, and S. Miyano (2008), Modeling nonlinear gene regulatory networks from time series gene expression data, *Journal of Bioinformatics and Computational Biology*, *6*(05), 961–979.

[25] Gao, X., D. Q. Pu, Y. Wu, and H. Xu (2012), Tuning parameter selection for penalized likelihood estimation of gaussian graphical model, *Statistica Sinica*, *22*(3), 1123.

[26] Gelman, A., and K. Shirley (2011), Inference from simulations and monitoring convergence, *Handbook of Markov chain Monte Carlo*, pp. 163–174.

[27] Giudici, P., and R. Castelo (2003), Improving markov chain monte carlo model search for data mining, *Machine learning*, *50*(1-2), 127–158.

[28] Green, P., and B. Silverman (1994), *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*, Chapman & Hall/CRC, London.

[29] Greenfield, A., A. Madar, H. Ostrer, and R. Bonneau (2010), DREAM4: combining genetic and dynamic information to identify biological networks and dynamical models, *PLoS ONE*, *5*(10), e13,397, doi:10.1371/journal.pone.0013397.

[30] Grzegorczyk, M., and D. Husmeier (2008), Improving the structure mcmc sampler for bayesian networks by introducing a new edge reversal move, *Machine Learning*, *71*(2-3), 265–305.

[31] Guckenheimer, J., and P. Holmes (1997), *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, Applied mathematical sciences ;v. 42, Springer, New York, xvi, 459 p.

[32] Gugushvili, S., and C. A. Klaassen (2012), $\sqrt{n}$-consistent parameter estimation for systems of ordinary differential equations: bypassing numerical integration via smoothing, *Bernoulli*, *18*(3), 1061–1098, doi:10.3150/11-BEJ362.

[33] Gustafsson, M., and M. Hörnquist (2009), Integrating various data sources for improved quality in reverse engineering of gene regulatory networks, *Handbook of Research on Computational Methodologies in Gene Regulatory Networks*, pp. 476–497.

[34] Hastie, T., and B. Efron (2012), *lars: Least Angle Regression, Lasso and Forward Stagewise*, r package version 1.1.

[35] Hastie, T., and R. J. Tibshirani (1990), *Generalized additive models*, Chapman & Hall, New York [etc.].

[36] Hastings, W. K. (1970), Monte carlo sampling methods using markov chains and their applications, *Biometrika*, *57*(1), 97–109.

[37] Hauser, A., and P. Bühlmann (2012), Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs, *The Journal of Machine Learning Research*, *13*(1), 2409–2464.

[38] Hauser, A., and P. Bühlmann (2012), Two optimal strategies for active learning of causal models from interventions, *Proc. of the 6th European Workshop on Probabilistic Graphical Models*, pp. 123–130.

[39] He, Y.-B., and Z. Geng (2008), Active learning of causal networks with intervention experiments and optimal designs, *Journal of Machine Learning Research*, *9*(11).

[40] Hecker, M., S. Lambeck, S. Toepfer, E. van Someren, and R. Guthke (2009), Gene regulatory network inference: Data integration in dynamic models—A review, *Biosystems*, *96*(1), 86–103, doi:10.1016/j.biosystems.2008.12.004.

[41] Heckerman, D., D. Geiger, and D. M. Chickering (1995), Learning bayesian networks: The combination of knowledge and statistical data, *Machine learning*, *20*(3), 197–243.

[42] Huang, J. Z., N. Liu, M. Pourahmadi, and L. Liu (2006), Covariance matrix selection and estimation via penalised normal likelihood, *Biometrika*, *93*(1), 85–98.

[43] Ideker, T. E., V. Thorsson, and R. M. Karp (2000), Discovery of regulatory interactions through perturbation: inference and experimental design, in *Pacific symposium on biocomputing*, vol. 5, pp. 302–313.

[44] Kalisch, M., and P. Bühlmann (2007), Estimating high-dimensional directed acyclic graphs with the pc-algorithm, *The Journal of Machine Learning Research*, *8*, 613–636.

[45] Kallenberg, O. (2002), *Foundations of modern probability*, Springer Science & Business Media.

[46] Lam, C., and J. Fan (2009), Sparsistency and rates of convergence in large covariance matrix estimation, *Annals of statistics*, *37*(6B), 4254.

[47] Lee, W.-P., and W.-S. Tzou (2009), Computational methods for discovering gene networks from expression data, *Briefings in Bioinformatics*, *10*(4), 408–423, doi:10.1093/bib/bbp028.

[48] Levina, E., A. Rothman, J. Zhu, et al. (2008), Sparse estimation of large covariance matrices via a nested lasso penalty, *The Annals of Applied Statistics*, *2*(1), 245–263.

[49] Madar, A., A. Greenfield, H. Ostrer, E. Vanden-Eijnden, and R. Bonneau (2009), The inferelator 2.0: A scalable framework for reconstruction of dynamic regulatory network models, in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 5448–5451.

[50] Madar, A., A. Greenfield, E. Vanden-Eijnden, and R. Bonneau (2010), DREAM3: network inference using dynamic context likelihood of relatedness and the inferelator, *PLoS ONE*, *5*(3), e9803, doi:10.1371/journal.pone.0009803.

[51] Madigan, D., J. York, and D. Allard (1995), Bayesian graphical models for discrete data, *International Statistical Review/Revue Internationale de Statistique*, pp. 215–232.

[52] Marbach, D., T. Schaffter, C. Mattiussi, and D. Floreano (2009), Generating Realistic In Silico Gene Networks for Performance Assessment of Reverse Engineering Methods, *Journal of Computational Biology*, *16*(2), 229–239, doi: 10.1089/cmb.2008.09TT, wingX.

[53] Marbach, D., R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, and G. Stolovitzky (2010), Revealing strengths and weaknesses of methods for gene network inference, *Proceedings of the National Academy of Sciences*, *107*(14), 6286–6291.

[54] Marbach, D., et al. (2012), Wisdom of crowds for robust gene network inference, *Nature Methods*, *9*(8), 796–804, doi:10.1038/nmeth.2016.

[55] Markowetz, F., and R. Spang (2007), Inferring cellular networks—a review, *BMC Bioinformatics*, *8*(Suppl 6), S5, doi:10.1186/1471-2105-8-S6-S5.

[56] Meinshausen, N., and P. Bühlmann (2006), High-dimensional graphs and variable selection with the lasso, *The Annals of Statistics*, pp. 1436–1462.

[57] Meister, A., Y. H. Li, B. Choi, and W. H. Wong (2013), Learning a nonlinear dynamical system model of gene regulation: A perturbed steady-state approach, *Annals of Applied Statistics*, *7*(3), 1311–1333.

[58] Michailidis, G. (2012), Statistical challenges in biological networks, *Journal of Computational and Graphical Statistics*, *21*(4), 840–855, doi: 10.1080/10618600.2012.738614.

[59] Molinelli, E., A. Korkut, W. Wang, M. Miller, N. Gauthier, and et al. (2013), Perturbation biology: Inferring signaling networks in cellular systems, *PLoS Comput Biol*, *9*.

[60] Murphy, K. (2001), Active learning of causal bayes net structure, *Technical Report*.

[61] Nakatsui, M., K. Horimoto, M. Okamoto, Y. Tokumoto, and J. Miyake (2010), Parameter optimization by using differential elimination: a general approach for introducing constraints into objective functions, *BMC Systems Biology*, *4*(Suppl 2), S9.

[62] Niinimäki, T., and M. Koivisto (2012), Annealed Importance Sampling for Structure Learning in Bayesian Networks, *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pp. 1579–1585.

[63] Niinimäki, T., P. Parviainen, and M. Koivisto (2012), Partial order mcmc for structure discovery in bayesian networks, *arXiv:1202.3753*.

[64] Oates, C. J., and S. Mukherjee (2012), Network inference and biological dynamics, *The Annals of Applied Statistics*, *6*(3), 1209–1235, doi:10.1214/11-AOAS532.

[65] Ong, I. M., J. D. Glasner, and D. Page (2002), Modelling regulatory pathways in E. coli from time series expression profiles, *Bioinformatics*, *18*(suppl 1), S241–S248, doi:10.1093/bioinformatics/18.suppl_1.S241.

[66] Parviainen, P., and M. Koivisto (2009), Exact structure discovery in bayesian networks with less space, in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 436–443, AUAI Press.

[67] Pearl, J. (2009), *Causality: models, reasoning, and inference*, second ed., Cambridge University Press, Cambridge, U.K. ; New York.

[68] Peters, J., J. Mooij, D. Janzing, and B. Schölkopf (2012), Identifiability of causal graphs using functional models, *arXiv preprint arXiv:1202.3757*.

[69] Pinna, A., N. Soranzo, and A. de la Fuente (2010), From knockouts to networks: Establishing direct cause-effect relationships through graph analysis, *PLoS ONE*, *5*(10), e12,912, doi:10.1371/journal.pone.0012912.

[70] Prill, R. J., D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky (2010), Towards a rigorous assessment of systems biology models: The DREAM3 challenges, *PLoS ONE*, *5*(2), e9202, doi:10.1371/journal.pone.0009202.

[71] Qi, X., and H. Zhao (2010), Asymptotic efficiency and finite-sample properties of the generalized profiling estimation of parameters in ordinary differential equations, *The Annals of Statistics*, *38*(1), 435–481.

[72] R Core Team (2012), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

[73] Ramsay, J. O., G. Hooker, D. Campbell, and J. Cao (2007), Parameter estimation for differential equations: a generalized smoothing approach, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *69*(5), 741–796, doi:10.1111/j.1467-9868.2007.00610.x.

[74] Ravikumar, P., J. Lafferty, H. Liu, and L. Wasserman (2009), Sparse additive models, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *71*(5), 1009–1030, doi:10.1111/j.1467-9868.2009.00718.x.

[75] Rothman, A. J., P. J. Bickel, E. Levina, J. Zhu, et al. (2008), Sparse permutation invariant covariance estimation, *Electronic Journal of Statistics*, *2*, 494–515.

[76] Schwarz, G., et al. (1978), Estimating the dimension of a model, *The annals of statistics*, *6*(2), 461–464.

[77] Shimamura, T., S. Imoto, R. Yamaguchi, A. Fujita, M. Nagasaki, and S. Miyano (2009), Recursive regularization for inferring gene networks from time-course gene expression profiles, *BMC Systems Biology*, *3*(1), 41.

[78] Shojaie, A., and G. Michailidis (2010), Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs, *Biometrika*, *97*(3), 519–538, doi:10.1093/biomet/asq038.

[79] Shojaie, A., A. Jauhiainen, M. Kallitsis, and G. Michailidis (2014), Inferring regulatory networks by combining perturbation screens and steady state gene expression profiles, *PLoS ONE*, *9*(2), e82,393, doi:10.1371/journal.pone.0082393.

[80] Sima, C., J. Hua, and S. Jung (2009), Inference of gene regulatory networks using time-series data: a survey, *Current genomics*, *10*(6), 416.

[81] Sloane, N. J. (2007), The on-line encyclopedia of integer sequences, in *Towards Mechanized Mathematical Assistants*, pp. 130–130, Springer.

[82] Spirtes, P., C. Glymour, and R. Scheines (2000), *Causation, Prediction, and Search. Adaptive Computation and Machine Learning*, MIT Press, Cambridge.

[83] Steinke, F., M. Seeger, and K. Tsuda (2007), Experimental design for efficient identification of gene regulatory networks using sparse bayesian models, *BMC systems biology*, *1*(1), 51.

[84] Tibshirani, R. (1996), Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288.

[85] Tong, S., and D. Koller (2001), Active learning for structure in bayesian networks, *ICJAI.*

[86] Van de Geer, S., P. Bühlmann, et al. (2013), ℓ_0-penalized maximum likelihood for sparse directed acyclic graphs, *The Annals of Statistics*, *41*(2), 536–567.

[87] Verma, T., and J. Pearl (1991), Equivalence and synthesis of causal models, in *Uncertainty in artificial intelligence*, vol. 6, p. 255.

[88] Voit, E. O. (1991), *Canonical nonlinear modeling: S-system approach to understanding complexity*, Van Nostrand Reinhold, New York.

[89] Voit, E. O. (2006), The intricate side of systems biology, *Proceedings of the National Academy of Sciences*, *103*(25), 9452–9457, doi:10.1073/pnas.0603337103.

[90] Wang, Y., T. Joshi, X.-S. Zhang, D. Xu, and L. Chen (2006), Inferring gene regulatory networks from multiple microarray datasets, *Bioinformatics*, *22*(19), 2413–2420.

[91] Wang, Y., R.-S. Wang, T. Joshi, D. Xu, X.-S. Zhang, L. Chen, and Y. Xia (2010), A linear programming framework for inferring gene regulatory networks by integrating heterogeneous data, *Computational Methodologies in Gene Regulatory Networks*, pp. 450–475.

[92] Yamaguchi, R., R. Yoshida, S. Imoto, T. Higuchi, and S. Miyano (2007), Finding module-based gene networks with state-space models - mining high-dimensional and short time-course gene expression data, *Signal Processing Magazine, IEEE*, *24*(1), 37–46, doi:10.1109/MSP.2007.273053.

[93] Yip, K. Y., R. P. Alexander, K.-K. Yan, and M. Gerstein (2010), Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data, *PLoS one*, *5*(1), e8121.

[94] Yuan, M., and Y. Lin (2007), Model selection and estimation in the gaussian graphical model, *Biometrika*, *94*(1), 19–35.