Bayesian network structure learning for the uncertain experimentalist

With applications to network biology

by

Daniel James Eaton

B.Sc., The University of British Columbia, 2005

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University Of British Columbia

June, 2007

Abstract

In this work, we address both the computational and modeling aspects of Bayesian network structure learning. Several recent algorithms can handle large networks by operating on the space of variable orderings, but for technical reasons they cannot compute many interesting structural features and require the use of a restrictive prior. We introduce a novel MCMC method that utilizes the deterministic output of the exact structure learning algorithm of Koivisto and Sood to construct a fast-mixing proposal on the space of DAGs. We show that in addition to fixing the order-space algorithms' shortcomings, our method outperforms other existing samplers on real datasets by delivering more accurate structure and higher predictive likelihoods in less compute time. Next, we discuss current models of intervention and propose a novel approach named the uncertain intervention model, whereby the targets of an intervention can be learned in parallel to the graph's causal structure. We validate our model experimentally using synthetic data generated from known ground truth. We then apply our model to two biological datasets that have been previously analyzed using Bayesian networks. On the T-cell dataset of Sachs et al. we show that the uncertain intervention model is able to better model the density of the data compared to previous techniques, while on the ALL dataset of Yeoh et al. we demonstrate that our method can be used to directly estimate the genetic effects of the disease.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I owe at least two sigmas of thank you density to my supervisor Dr. Kevin Murphy. Over the past two years he has been an incredible source of inspiration and motivation, keeping the Bayes ball rolling even in the face of occasionally disappointing results or formidable bugs. I found the support of an NSERC scholarship to be most beneficial, and therefore thank Canada for supporting my work.

I also appreciate the instruction of several professors in the Computer Science department, and the help I received from peers while preparing papers for conference submission, especially Mark Schmidt, Hoyt Koepke and Hendrik Kueck.

I would like to acknowledge my family's contributions: Joan valiantly proofread my thesis, while Sheena, Doug and Sarah provided me with motivation to graduate on time, some in more gentle ways then others.

I also wish to thank Michael Forbes for generously sharing his UBC thesis LaTeX template; it made formatting a breeze.

Daniel Eaton

University of British Columbia
June 2007

Dedication

This thesis is dedicated to my parents, *Doug* and *Joan*, in recognition of their continuous support through each episode of my academic career.

Part I

Thesis

Chapter 1

Introduction

1.1 Motivation

Bayesian networks are a powerful tool for modeling stochastic systems, as they enable high-dimensional joint probability distributions to be represented efficiently. The model is especially popular due to the availability of off-the-shelf algorithms and software to learn them from data, and then to extract meaningful quantities in support of scientific hypotheses. More importantly, unlike undirected models, the inter-variable relationships encoded by a Bayesian network can be interpreted causally, meaning that the notion of "smoking causes cancer" can be characterized directly, rather than simply, "smoking is linked to cancer". This distinction may appear subtle, but the notion of causality is at the heart of modern science [45]. Therefore, it comes as no surprise that Bayesian networks are rapidly spreading beyond the domains for which they were originally created.

The systems biology community has recently begun exploiting the model to analyze high-throughput biological data, such as gene expression microarrays [12, 20, 25]. In many classical applications, the relationships between variables were assumed known *a priori*; in other words, the Bayesian network structure was given, perhaps by a human expert. The goal then became to learn the associated model parameters. In contrast, many system biologists aim to learn the structure of the network from data, for example to infer gene regulatory interactions, discover new drugs, or to guide costly future experiments. Evidence suggests that cancers and many other diseases manifest themselves by disrupting specific cellular processes [1]. Gene expression microarrays can measure tran-

scription levels for thousands of genes in a cell, offering insight into the complex cell dynamics that give rise to diseases.

Causal Bayesian network models supply a natural means of modeling systems biology data, potentially yielding answers about cellular processes, or identifying the effects of external agents, such as diseases or drugs. In order to uniquely infer causal relationships, it is necessary to perform experiments where variables are perturbed or manipulated to particular known values. In biological applications, an intervention might be accomplished by injecting chemicals into a cell or performing a specific gene knockout. Alternatively, nature provides unwanted perturbations in the form of disease, which can be construed as virtual experiments.

Unfortunately, even if both observational and experimental data is readily available, Bayesian network structure learning remains a challenging problem, largely due to the super-exponential growth of the space of valid networks given the number of variables. Valid structures are represented by directed acyclic graphs (DAGs), and there are $O(d!2^{\binom{d}{2}})$ DAGs on d nodes [47]. Table 1.1 shows the growth for small d. Exhaustive search or naive marginalization over such a space is clearly intractable in general. Therefore, without clever algorithms it is necessary to resort to local search through a huge hypothesis space.

d	2	3	4	5	6	7	8	9
#DAG(d)	3	25	543	29281	3781503	1.1e9	7.8e11	1.2e15

Table 1.1: Number of DAGs on d nodes

The scarcity of data inherent to most biological applications greatly exacerbates this computational problem. Often, the number of variables dwarfs the sample size, meaning that many DAGs are likely to fit the data well. In this setting, it would be dangerous to commit to a particular structure and make any interpretation on the causal relationships between variables, since there conceivably could be another structure that also fits the data well, and leads to contradictory conclusions. Full Bayesian model averaging would be the greatly

preferable strategy; however, the naive approach becomes intractable for more than a mere 6 variables.

Lastly, many of the scientific questions that the systems biology community poses are unanswerable using existing models of intervention. Typically when a system is intervened on, the experimenter assumes the targets of their perturbation are known, and exploits this knowledge in the prescribed way [8] during structure learning. The perturbation may instead have a "fat hand" and cause unexpected side-effects that cannot be explained by the intrinsic relationships between variables. Perhaps the goal is to learn the intervention's targets, for example to measure the side-effects of a new treatment. Unfortunately, it is not possible to directly answer these, nor other important questions with existing models.

1.2 Outline of Thesis

This thesis presents novel algorithms and models for Bayesian network structure learning aimed at overcoming the challenges introduced in Section 1.1. In Chapter 2, we discuss existing computational methods, including Markov chain monte carlo (MCMC) and exact Bayesian model averaging (BMA), and briefly discuss their shortcomings. Next, we introduce an algorithm that combines MCMC with exact BMA to solve them, and show that it simultaneously outperforms the other sampling methods based on time and accuracy. Chapter 3 presents a new model of experimental data, allowing for uncertainty in the targets of interventions. The new model is verified on synthetic data, and then tested on two gene expression datasets, producing results that agree with other reported analyses, but also providing novel benefits that the other methods cannot. The chapter also shows how to adapt a recent exact structure learning algorithm to handle experimental data.

Chapters 2 (based on [13]) and 3 (based on [14]) are structured in a self-contained format, each containing background, results and conclusions relevant to their respective topics. However, it must be noted that each topic is heavily

intertwined: algorithms borrowing models to test on data, and models using algorithms to do computation.

Chapter 2

Structure learning by dynamic programming and MCMC

2.1 Introduction

Directed graphical models are useful for a variety of tasks, ranging from density estimation to scientific discovery. One of the key challenges is to learn the structure of these models from data. Often (e.g., in molecular biology) the sample size is quite small relative to the size of the hypothesis space. In such cases, the posterior over graph structures given data, $p(G|D)$, gives support to many possible models, and using a point estimate (such as MAP) could lead to unwarranted conclusions about the structure, as well as poor predictions about future data. It is therefore preferable to use Bayesian model averaging. If we are interested in the probability of some structural feature f (e.g., $f(G) = 1$ if there is an edge from node i to j and $f(G) = 0$ otherwise), we can compute the posterior mean estimate $E(f|D) = \sum_G f(G)p(G|D)$. Similarly, to predict future data, we can compute the posterior predictive distribution $p(x|D) = \sum_G p(x|G)p(G|D)$.

Since there are $O(d!2^{\binom{d}{2}})$ DAGs (directed acyclic graphs) on d nodes [47], exact Bayesian model averaging is intractable in general. However, if the model space is restricted special cases arise where full averaging is possible; for example, over all trees [40] or all graphs consistent with a given node ordering [10]. If

the ordering is unknown, we can use MCMC techniques to sample orders, and sample DAGs given each such order [19, 21, 29]. However, Koivisto and Sood [31, 32] showed that one can use dynamic programming (DP) to marginalize over orders analytically. This technique enables one to compute all marginal posterior edge probabilities, $p(G_{ij} = 1|D)$, exactly in $O(d2^d)$ time. Although exponential in d, this technique is quite practical for $d \leq 20$, and is much faster than comparable MCMC algorithms on similar sized problems[1].

Unfortunately, the DP method has three fundamental limitations, even for small domains. The first problem is that it can only be used with certain kinds of graph priors which satisfy a "modularity" condition, which will be described in Section 2.3. Although this seems like a minor technical problem, it can result in significant bias. This can lead to unwarranted conclusions about structure as well as poor predictive performance, even in the large sample setting. The second problem is that it can only compute posteriors over modular features; thus it cannot be used to compute the probability of features like "is there a path between nodes i and j via k", or "is i an ancestor of j". Such long-distance features are often of more interest than direct edges. The third problem is that it is expensive to compute predictive densities, $p(x|D)$. Since the DP method integrates out the graph structures, it has to keep all the training data D around, and predict using $p(x|D) = p(x, D)/p(D)$. Both terms can be computed exactly using DP, but this requires re-running DP for each new test case x. In addition, since the DP algorithm assumes complete data, if x is incompletely observed (e.g., we want to "fill in" some of it), we must run the DP algorithm potentially an exponential number of times. For the same reason, we cannot sample from $p(x|D)$ using the DP method.

We propose to fix all three of these shortcomings by combining DP with the Metropolis-Hastings (MH) algorithm. The basic idea is simply to use the

[1] Our Matlab/C implementation takes 1 second for $d = 10$ nodes and 6 minutes for $d = 20$ nodes on a standard laptop. The cost is dominated by the marginal likelihood computation, which all algorithms must perform. Our code is freely available; please see Appendix A for instructions to obtain it.

DP algorithm as an informative (data driven) proposal distribution for moving through DAG space, thereby getting the best of both worlds: a fast deterministic approximation, plus unbiased samples from the correct posterior, $G^s \sim p(G|D)$. These samples can then be used to compute the posterior mean of arbitrary features, $E[f|D] \approx \frac{1}{S}\sum_{s=1}^{S} f(G^s)$, or the posterior predictive distribution, $p(x|D) \approx \frac{1}{S}\sum_{s=1}^{S} p(x|G^s)$. Results presented in Section 2.5 show that this hybrid method produces more accurate estimates than other approaches, given a comparable amount of compute time.

The idea of using deterministic algorithms as a proposal has been explored before e.g. [11], but not, as far as we know, in the context of graphical model structure learning. Further, in contrast to [11], our proposal is based on an exact algorithm rather than an approximate algorithm.

2.2 Previous work

The most common approach to estimating the posterior $p(G|D)$, or marginal features thereof, is to use the Metropolis-Hastings (MH) algorithm, using a proposal that randomly adds, deletes or reverses an edge; this has been called MC^3 for Markov Chain Monte Carlo Model Composition [36]. (See also [24] for some improvements, and [35] for a related approach called Occam's window.) Unfortunately, this proposal is very local, and the resulting chains do not mix well in more than about 10 dimensions. An alternative is to use Gibbs sampling on the adjacency matrix [37]. In our experience, this gets "stuck" even more easily, although this can be ameliorated somewhat by using multiple restarts, as the experimental results will demonstrate.

A different approach, first proposed in [21], is to sample in the space of node orderings using MH. It is based on the fact that conditioned on a node ordering $\prec$, the probability of the data and a feature factorizes

$$p(X, f|I, \prec) = \prod_{i=1}^{d} \sum_{G_i \subseteq U_i^{\prec}} p(G_i|U_i) p(X_i|X_{G_i}, I) f_i(G_i) \tag{2.1}$$

where $p(x, f)$ means $p(x, f(G) = 1)$ and $U_i^{\prec} = \{j : j \prec i\}$ are the set of nodes that preceed i. This fact was first observed by [3], and was exploited by [21] using a MH proposal that randomly swaps the ordering of nodes. For example,

$$(1, 2, 3, 4, 5, 6) \rightarrow (1, 5, 3, 4, 2, 6)$$

where we swapped 2 and 5. This is a smaller space ("only" $O(d!)$), and is "smoother", allowing chains to mix more easily. [21] provides experimental evidence that this approach gives much better results than MH in the space of DAGs with the standard add/ delete/ reverse proposal. Unfortunately, in order to use this method, one is forced to use a modular prior, which has various undesirable consequences that we discuss in Section 2.3. Ellis and Wong [19] realized this, and suggested using an importance sampling correction. However, computing the exact correction term is #P-hard, and our empirical results suggest that their approximate correction yields inferior structure learning and predictive density compared to our method.

An alternative to sampling orders is to analytically integrate them out using dynamic programming (DP) [31, 32]. The algorithm is complex, but the key idea can be stated simply: when considering different variable orderings — say $(3, 2, 1)$ and $(2, 3, 1)$ — the contribution to the marginal likelihood for some nodes can be re-used. For example, $p(X_1|X_2, X_3)$ is the same as $p(X_1|X_3, X_2)$, since the order of the parents does not matter. By appropriately caching terms, one can devise an $O(d2^d)$ algorithm to exactly compute the marginal likelihood and marginal posterior features. The inputs to this algorithm are a modular prior (see Section 2.3) and the local conditional marginal likelihoods, which must be computed for every node i and every possible parent set G_i (up to size k). There are $\sum_{p=0}^{k} \binom{d}{p} = O(d^k)$ such terms, therefore the time complexity of the DP algorithm including marginal likelihood computation is $O(d2^d + d^{k+1}C(N))$. Here, $C(N)$ is the amount of time needed to compute each marginal likelihood term as a function of the sample size N. In practise, the polynomial term usually strongly dominates the exponential term.

To compute the posterior predictive density, $p(x|D)$, the standard approach

is to use a plug-in estimate $p(x|D) \approx p(x|\hat{G}(D))$. Here $\hat{G}$ may be an approximate MAP estimate computed using local search [27], or the MAP-optimal DAG which can be found by the recent algorithm of [50] (which takes $o(d^2 2^{d-2})$ time.) Alternatively, $\hat{G}$ could be a tree; this is a popular choice for density estimation since one can compute the optimal tree structure in $O(d^2 \log d)$ time [7, 41].

It can be proven that averaging over the uncertainty in G will, on average, produce higher test-set predictive likelihoods [34]. The DP algorithm can compute the marginal likelihood of the data, $p(D)$ (marginalizing over all DAGs), and hence can compute $p(x|D) = p(x, D)/p(D)$ by calling the algorithm twice. (We only need the "forwards pass" of [32], using the feature $f = 1$; we do not need the backwards pass of [31].) However, this is very expensive, since we need to compute the local marginal likelihoods for every possible family on the expanded data set for every test case x. In Section 2.5 we will show that by averaging over a sample of graphs our method gives comparable predictive performance at a much lower cost.

2.3 Modular priors

Some of the best current methods for Bayesian structure learning operate in the space of node orders rather than the space of DAGs, either using MCMC [19, 21, 29] or dynamic programming [31, 32]. Rather than being able to define an arbitrary prior on graph structures $p(G)$, methods that work with orderings define a joint prior over graphs G and orders $\prec$ as follows:

$$p(\prec, G) = \frac{1}{Z} \prod_{i=1}^{d} q_i(U_i^{\prec}) \rho_i(G_i) \times I(\text{consistent}(\prec, G))$$

where $U_i^{\prec}$ is the set of predecessors (possible parents) for node i in $\prec$, and G_i is the set of actual parents for node i. We say that a graph structure $G = (G_1, \ldots, G_d)$ is consistent with an order $(U_1, \ldots, U_d)$ if $G_i \subseteq U_i$ for all i. (In addition we require that G be acyclic, so that $\prec$ exists.) Note that U_i and G_i are not independent. Thus the q_i and ρ_i terms can be thought of as factors

or constraints, which define the joint prior $p(\prec, G)$. This is called a modular prior, since it decomposes into a product of local terms. It is important for computational reasons that $\rho_i(G_i)$ only give the prior weight to *sets* of parents, and not to their relative order, which is determined by $q_i(U_i)$. This feature is what enables the order space algorithms to re-use scores for all orderings of a parent set, and turn a sum over the super-exponential structure space into a sum over the factorial order space, and then further reduce it to an exponential complexity.

From the joint prior, we can infer the marginal prior over graphs, $p(G) = \sum_{\prec} p(\prec, G)$. Unfortunately, this prior favors graphs that are consistent with more orderings. For example, the fully disconnected graph is the most probable under a modular prior, and trees are more probable than chains, even if they are Markov equivalent (e.g., $1 \leftarrow 2 \rightarrow 3$ is more probable than $1 \rightarrow 2 \rightarrow 3$). This can cause problems for structural discovery. To see this, suppose the sample size is very large, so the posterior concentrates its mass on a single Markov equivalence class. Unfortunately, the effects of the prior are not "washed out", since all graphs with the equivalence class have the same likelihood. Thus we may end up predicting that certain edges are present due to artifacts of our prior, which was merely chosen for technical convenience.

In the absence of prior knowledge, one may want to use a uniform prior over DAGs[2]. However, this cannot be encoded as a modular prior. To see this, let us use a uniform prior over orderings, $q_i(U_i) = 1$, so $p(\prec) = 1/(d!)$. This is reasonable since typically we do not have prior knowledge on the order. For the parent factors, let us use $\rho_i(G_i) = 1$; we call this a "modular flat" prior. However, this combination is not uniform over DAGs after we sum over orderings: see Figure 2.1. A more popular alternative (used in [19, 21, 31, 32]) is to take $\rho_i(G_i) \propto \binom{d-1}{|G_i|}^{-1}$; we call this the "Koivisto" prior. This prior says that different cardinalities of parents are considered to be equally likely a priori.

[2] One could argue that we should use a uniform prior over PDAGs, but we will often be concerned with learning causal models from interventional data, in which case we have to use DAGs.

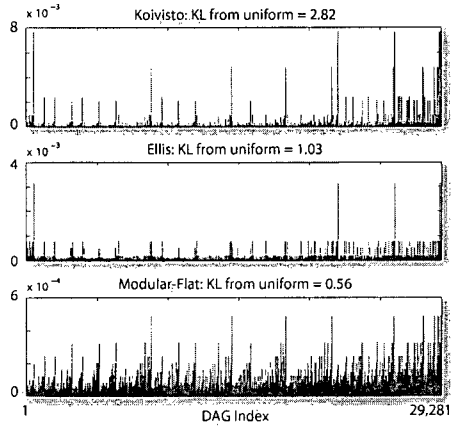


Figure 2.1: **Deviation of various modular priors from uniform.** Priors on all 29,281 DAGs on 5 nodes. Koivisto prior means using $\rho_i(G_i) \propto \binom{d-1}{|G_i|}^{-1}$. Ellis prior means the same ρ_i, but dividing by the number of consistent orderings for each graph (computed exactly). Modular flat means using $\rho_i(G_i) \propto 1$. This is the closest to uniform in terms of KL distance, but will still introduce artifacts. If we use $\rho_i(G_i) \propto 1$ and divide by the number of consistent orders. we will get a uniform distribution, but computing the number of consistent orderings is #P-hard.

However, the resulting $p(G)$ is even further from uniform: see Figure 2.1.

Ellis and Wong [19] recognized this problem, and tried to fix it as follows. Let $p^*(G) = \frac{1}{Z}\prod_i \rho_i^*(G)$ be the desired prior, and let $p(G)$ be the actual modular prior implied by using ρ_i^* and $q_i = 1$. We can correct for the bias by using an importance sampling weight given by

$$w(G) = \frac{p^*(G)}{p(G)} = \frac{\frac{1}{Z}\prod_i \rho_i^*(G_i)}{\sum_\prec \frac{1}{Z}\prod_i \rho_i^*(G_i) I(\text{consistent}(\prec, G))}$$

If we set $\rho_i^* = 1$ (the modular flat prior), then this becomes

$$w(G) = \frac{1}{\sum_\prec I(\text{consistent}(\prec, G)}$$

Thus this weighting term compensates for overcounting certain graphs, and induces a globally uniform prior, $p(G) \propto 1$. However, computing the denominator (the number of orders consistent with a graph) is #P-complete [2]. Ellis and Wong approximated this sum using the sampled orders, $w(G) \approx \frac{1}{\sum_{s=1}^S I(\text{consistent}(\prec^s, G))}$. However, these samples $\prec^s$ are drawn from the posterior $p(\prec | D)$, rather than the space of all orders, so this is not an unbiased estimate. Also, they used $\rho_i(G_i) \propto \binom{d-1}{|G_i|}^{-1}$, rather than $\rho_i = 1$, which still results in a highly non-uniform prior, even after exact reweighting (see Figure 2.1). In contrast, our method can cheaply generate samples from an arbitrary prior.

2.4 Our method

As mentioned above, our method is to use the Metropolis-Hastings algorithm with a proposal distribution that is a mixture of the standard local proposal, that adds, deletes or reverses an edge at random, and a more global proposal that uses the output of the DP algorithm:

$$q(G'|G) = \begin{cases} q_{local}(G'|G) & \text{w.p. } \beta \\ q_{global}(G') & \text{w.p. } 1-\beta. \end{cases}$$

The local proposal chooses uniformly at random from all legal *single edge* additions, deletions and reversals. Let $\text{nbd}(G)$ denote the set of acyclic neighbors

generated in this way. We have the proposal distribution

$$q_{local}(G'|G) = \frac{1}{|\text{nbd}(G)|} I(G' \in \text{nbd}(G)),$$

and accept moves proposed from $q_{local}(G'|G)$ with probability

$$\alpha_{local} = \min\left(1, \frac{p(D|G')p(G')}{p(D|G)p(G)} \frac{|\text{nbd}(G)|}{|\text{nbd}(G')|}\right).$$

The global proposal includes an edge between i and j with probability $p_{ij} + p_{ji} \leq 1$, where $p_{ij} = p(G_{ij}|D)$ are the exact marginal posteriors computed using DP (using a modular prior). If this edge is included, it is oriented as $i \rightarrow j$ w.p. $q_{ij} = p_{ij}/(p_{ij} + p_{ji})$, otherwise it is oriented as $i \leftarrow j$. After sampling each edge pair, we check if the resulting graph is acyclic. (The acyclicity check can be done in amortized constant time using the ancestor matrix trick [24].) This leads to

$$q_{global}(G') = \left(\prod_i \prod_{j>i} (p_{ij} + p_{ji})^{I(G'_{ij} + G'_{ji} > 0)}\right) \times \left(\prod_{ij} q_{ij}^{I(G'_{ij}=1)}\right) I(\text{acyclic}(G')).$$

We then accept moves proposed from $q_{global}(G')$ with probability

$$\alpha_{global} = \min\left(1, \frac{p(D|G')p(G')}{p(D|G)p(G)} \frac{q_{global}(G)}{q_{global}(G')}\right).$$

If we set $\beta = 1$, we get the standard local proposal. If we set $\beta = 0$, we get a purely global proposal. Note that $q_{global}(G')$ is independent of G, so this is an independence sampler. We tried various other settings of β (including adapting it according to a fixed schedule), which results in performance somewhere in between purely local and purely global.

For $\beta > 0$ the chain is aperiodic and irreducible, since the local proposal has both properties [36, 46]. However, if $\beta = 0$, the chain is not necessarily aperiodic and irreducibile, since the global proposal may set $p_{ij} = p_{ji} = 0$. This problem is easily solved by truncating edge marginals which are too close to 0 or 1, and making the appropriate changes to $q_{global}(G')$. Specifically, any $p_{ij} < C$ is set to C, while $p_{ij} > 1 - C$ are set to $1 - C$. We used $C = 1e - 4$ in our experiments.

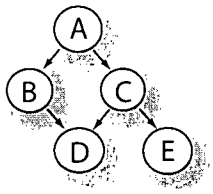


Figure 2.2: **Cancer network.** First reported in [22].

2.4.1 Likelihood models

For simplicity, we assume all the conditional probability distributions (CPDs) are multinomials (tables), $p(X_i = k|X_{G_i} = j, \theta) = \theta_{ijk}$, though our software (see Appendix A) can handle the linear-Gaussian case as well. We make the usual assumptions of parameter independence and modularity [27], and we use uniform conjugate Dirichlet priors $\theta_{ij} \sim Dir(\alpha_i, \ldots, \alpha_i)$, where we set $\alpha_i = 1/(q_i r_i)$, where q_i is the number of states for node X_i and r_i is the number of states for the parents X_{G_i}. The resulting marginal likelihood,

$$\begin{aligned} p(D|G) &= \prod_i p(X_i|X_{G_i}) \\ &= \prod_i \int [\prod_n p(X_{n,i}|X_{n,G_i}, \theta_i)] p(\theta_i|G_i) d\theta_i \end{aligned}$$

can be computed in closed form, and is called the BDeu (Bayesian Dirichlet likelihood equivalent uniform) score [27]. We use AD trees [43] to compute these terms efficiently. Note that our technique can easily be extended to other CPDs (e.g., decision trees [5]), provided $p(X_i|X_{G_i})$ can be computed or approximated (e.g., using BIC).

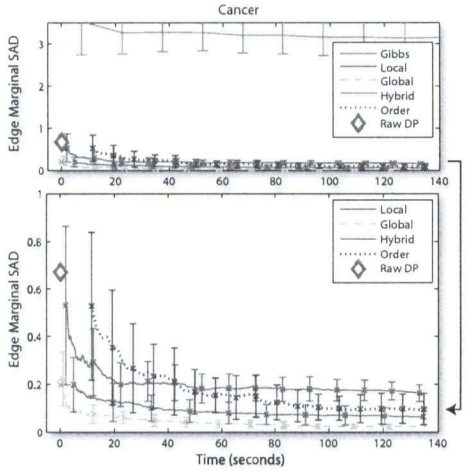


Figure 2.3: **Convergence to edge marginals on Cancer network.** SAD error vs running time on the 5 node Cancer network (shown in Figure 2.2). The Gibbs sampler performs poorly, therefore we replot the graph with it removed (bottom figure). Note that 140 seconds corresponds to about 130,000 samples from the hybrid sampler. The error bars (representing one standard deviation across 25 chains starting in different conditions) are initially large, because the chains have not burned in. This figure is best viewed in colour.

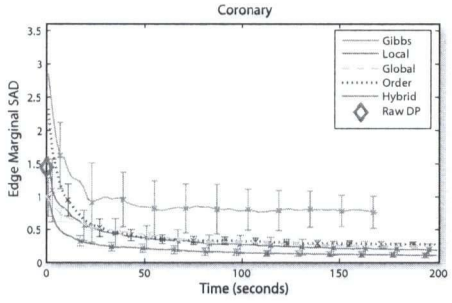


Figure 2.4: **Convergence to edge marginals on CHD dataset.** Similar to Figure 2.3, but on the 6 node CHD dataset.

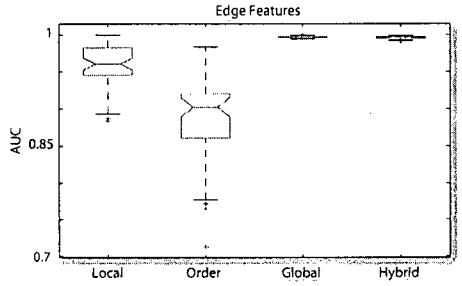


Figure 2.5: **Edge recovery performance of MCMC methods on Child network.** Area under the ROC curve (averaged over 10 MCMC runs) for detecting edge presence for different methods on the $d = 20$ node Child network with $n = 10k$ samples using 200 seconds of compute time. The AUC of the exact DP algorithm is indistinguishable from the global method and hence is not shown.

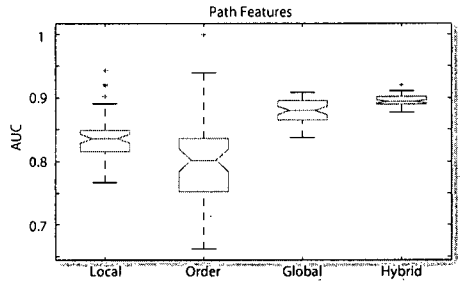


Figure 2.6: **Path recovery performance of MCMC methods on Child network.** Area under the ROC curve (averaged over 10 MCMC runs) for detecting path presence for different methods on the $d = 20$ node Child network with $n = 10k$ samples using 200 seconds of compute time.

2.5 Experimental results

2.5.1 Speed of convergence to the exact posterior marginals

In this section we compare the accuracy of different algorithms in estimating $p(G_{ij} = 1|D)$ as a function of their running time, where we use a uniform graph prior $p(G) \propto 1$. (Obviously we could use any other prior or feature of interest in order to assess convergence speed, but this seemed like a natural choice, and enables us to compare to the raw output of DP.) Specifically, we compute the sum of absolute differences (SAD), $S_t = \sum_{ij} |p(G_{ij} = 1|D) - q_t(G_{ij} = 1|D)|$, versus running time t, where $p(G_{ij} = 1|D)$ are the exact posterior edge marginals (computed using brute force enumeration over all DAGs) and $q_t(G_{ij}|D)$ is the approximation based on samples up to time t. We compare 5 MCMC methods: Gibbs sampling on elements of the adjacency matrix, purely local moves through DAG space ($\beta = 1$), purely global moves through DAG space using the DP proposal ($\beta = 0$, which is an independence sampler), a mixture of local and global (probability of local move is $\beta = 0.1$), and an MCMC order sampler [21] with Ellis' importance weighting term.[3] (In the figures, these are called as follows: $\beta = 1$ is "Local", $\beta = 0$ is "Global", $\beta = 0.1$ is "Hybrid".) In our implementation of the order sampler, we took care to implement the various caching schemes described in [21], to ensure a fair comparison. However, we did not use the sparse candidate algorithm or any other form of pruning.

For our first experiment, we sampled data from the 5 node "Cancer network" of [22] (shown in Figure 2.2) and then ran the different methods. In Figure 2.3, we see that the DP+MCMC samplers outperform the other samplers. We also ran each method on the well-studied coronary heart disease (CHD) dataset [18]. This consists of about 200 cases of 6 binary variables, encoding such things as "is your blood pressure high?", "do you smoke?", etc. In Figure 2.4, we see

[3] Without the reweighting term, the MCMC order sampler [21] would eventually converge to the same results (as measured by SAD) as the DP method [31, 32].

again that our DP+MCMC method is the fastest and the most accurate.

2.5.2 Structural discovery

In order to assess the scalability of our algorithm, we next looked at data generated from the 20 node "Child" network used in [55] (shown in Figure 2.7). We sampled $n = 10,000$ records using random multinomial CPDs sampled from a Dirichlet, with hyper-parameters chosen by the method of [6], which ensures strong dependencies between the nodes[4]. Stronger dependencies increase the likelihood that the distribution will be numerically faithful to the conditional independency assumptions encoded by the structure. Next, we compute the posterior over two kinds of features: edge features, $f_{ij} = 1$ if there is an edge between i and j (in either orientation), and path features, $f_{ij} = 1$ if there is a directed path from i to j. (Note that the latter cannot be computed by DP; to compute it using the order sampler of [21] requires sampling DAGs given an order.) We can no longer compare the estimated posteriors to the exact posteriors (since $d = 20$), but we can compare them to the ground truth values from the generating network. Following [28, 31], we threshold these posterior features at different levels, to trade off sensitivity and specificity. We summarize the resulting ROC curves in a single number, namely area under the curve (AUC).

The results for edge features are shown in Figure 2.5. We see that the DP+MCMC methods do very well at recovering the true undirected skeleton of the graph, obtaining an AUC of 1.0 (same as the exact DP method). We see that our DP+MCMC samplers are significantly better (at the 5% level) than the DAG sampler and the order sampler. The order sampler does not do as well as the others, for the same amount of run time, since each sample is more expensive to generate.

The results for path features are shown in Figure 2.6. Again we see that

[4] For example, consider a node ℓ with 3 states and 4 parent states. The method of [6] prescribes that we pick a "basis vector" $(1, 1/2, 1/3)$, and then, for the j'th parent state, we sample $\theta_{ij\cdot} \sim Dir(s\alpha_{ij\cdot})$, where $\alpha_{i1\cdot} \propto (1, 1/2, 1/3)$, $\alpha_{i2\cdot} \propto (1/2, 1/3, 1)$, $\alpha_{i3\cdot} \propto (1/3, 1, 1/2)$, $\alpha_{i4\cdot} \propto (1, 1/2, 1/3)$, and $s = 10$ is an effective sample size.

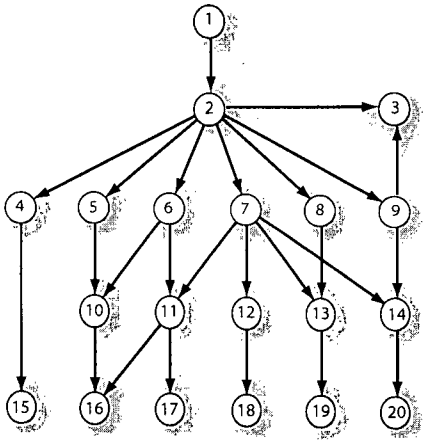


Figure 2.7: **Child network.** First reported in [9].

the DP+MCMC method (using either $\beta = 0$ or $\beta = 0.1$) yields statistically significant improvement (at the 5% level) in the AUC score over other MCMC methods on this much harder problem.

2.5.3 Accuracy of predictive density

In this section, we compare the different methods in terms of the log loss on a test set:

$$\ell = E \log p(x|D) \approx \frac{1}{m} \sum_{i=1}^{m} \log p(x_i|D)$$

where m is the size of the test set and D is the training set. This is the ultimate objective test of any density estimation technique, and can be applied to any dataset, even if the "ground truth" structure is not known. The hypothesis that we wish to test is that methods which estimate the posterior $p(G|D)$ more accurately will also perform better in terms of prediction. We test this hypothesis on three datasets: synthetic data from a 15-node network, the "Adult" US census dataset from the UC Irvine repository and a biological dataset related to the human T-cell signalling pathway [48].

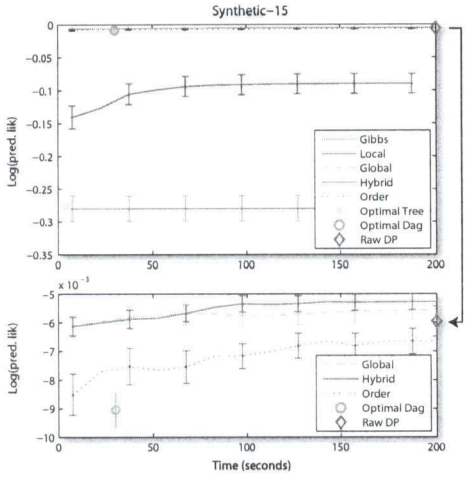


Figure 2.8: **Test set log likelihood vs training time on Synthetic-15 network.** $d = 15$, $N = 1500$. The bottom figure presents the "good" algorithms in higher detail by removing the poor performers. Results for the factored model are an order of magnitude worse and therefore not plotted. Note that the DP algorithm actually took over *two hours* to compute.

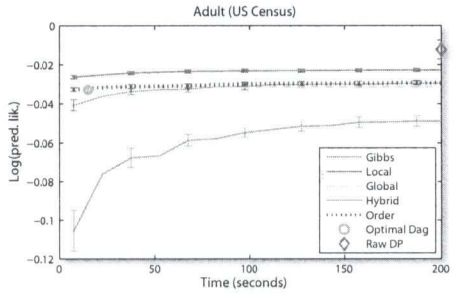


Figure 2.9: **Test set log likelihood vs training time on Adult dataset.** $d = 14$, $N = 49k$. DP algorithm actually took over *350 hours* to compute. The factored and maximum likelihood tree results are omitted since they are many orders of magnitude worse and ruin the graph's vertical scale.

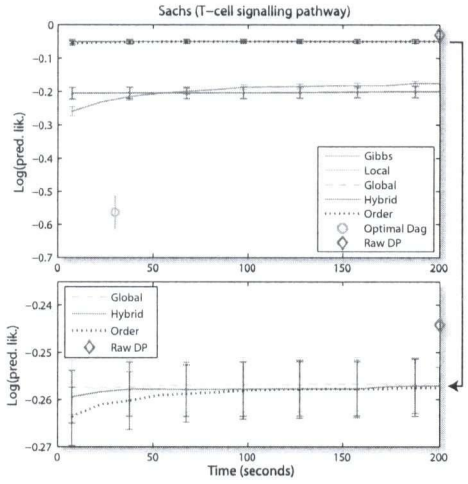


Figure 2.10: **Test set log likelihood vs training time on T-cell dataset.** $d = 11$, $N = 5400$. DP algorithm actually took over *90 hours* to compute. The factored model and optimal tree plugins are again omitted for clarity. The bottom part of the figure is a zoom in of the best methods.

In addition to DP and the MCMC methods mentioned above, we also measured the performance of plug-in estimators consisting of: a fully factorized model (the disconnected graph), the maximum likelihood tree [7], and finally the MAP-optimal DAG gotten from the algorithm of [50]. We measure the likelihood of the test data as a function of *training time*, $\ell(t)$. That is, to compute each term in ℓ we use $p(x_i|D) = \frac{1}{S_t}\sum_{s=1}^{S_t} p(x_i|G^s)$, where S_t is the number of samples that can be computed in t seconds. Thus a method that mixes faster should produce better estimates. Note that, in the Dirichlet-multinomial case, we can quickly compute $p(x|G^s)$ by plugging in the posterior mean parameters:

$$p(x|G^s) = \prod_{ijk} \overline{\theta}_{ijks}^{I(x_i=j, x_{G_i}=k)}$$

where $\overline{\theta}_{ijks} = E[\theta_{ijk}|D, G^s]$. If we have missing data, we can use standard Bayes net inference algorithms to compute $p(x|G^s, \overline{\theta})$.

In contrast, for DP, the "training" cost is computing the normalizing constant $p(D)$, and the test time cost involves computing $p(x_i|D) = p(x_i, D)/p(D)$ for each test case x_i separately. Hence we must run the DP algorithm m times to compute ℓ (each time computing the marginal likelihoods for all families on the augmented data set x_i, D). DP is thus similar to a non-parametric method in that it must keep around all the training data, and is expensive to apply at run-time. This method becomes even slower if x is missing components: suppose k binary features are missing, then we have to call the algorithm 2^k times to compute $p(x|D)$.

For the first experiment, we generated several random networks, sampling the nodes' arities uniformly at random from between 2-4 and the parameters from a Dirichlet. Next, we sampled $100d$ records (where d is the number of nodes) and performed 10-fold cross-validation. Here, we just show results for a 15-node network, which is representative of the other synthetic cases. Figure 2.8 plots the mean predictive likelihood across cross-validation folds and 5 independent sampler runs against training time. On the zoomed plot at the bottom, we can see that the hybrid and global MCMC methods are significantly better than order sampling. Furthermore, they seem to be better than

exact DP, which is perhaps being hurt by its modular prior. All of these Bayes model averaging (BMA) methods (except Gibbs) significantly beat the plugin estimators, including the MAP-optimal structure.

In the next experiment we used the "Adult" US census dataset, which consists of 49,000 records with 14 attributes, such as "education", "age", etc. We use the discretized version of this data as previously used in [42]. The average arity of the variables is 7.7. The results are shown in Figure 2.9. The most accurate method is DP, since it does exact BMA (although using the modular prior), but it is also the slowest. Our DP+MCMC method (with $\beta = 0.1$) provides a good approximation to this at a fraction of the cost (it took over 350 hours to compute the predictive likelihood using the DP algorithm). The other MH methods also do well, while Gibbs sampling does less well. The plug-in DAG is not as good as BMA, and the plug-in Chow-Liu tree and plug-in factored model do so poorly on this dataset that their results are not shown (lest they distort the scale). (These results are averaged over 10 MCMC runs and over 10 cross-validation folds.)

Finally, we applied the method to a biological data set [48] which consists of 11 protein concentration levels measured (using flow cytometry) under 6 different interventions, plus 3 unperturbed measurements. 600 measurements are taken in each condition yielding a total dataset of $N = 5400$ records. Sachs et al. discretized the data into 3 states, and we used this version of the data. We modified the marginal likelihood computations to take into account the interventional nature of the data as in [8]. The results are shown in Figure 2.10. Here we see that DP gives the best result, but takes 90 hours. The global, hybrid and order samplers all do almost as well at a fraction of the cost. The local proposal and Gibbs sampling perform about equally. All methods that perform BMA beat the optimal plugin.

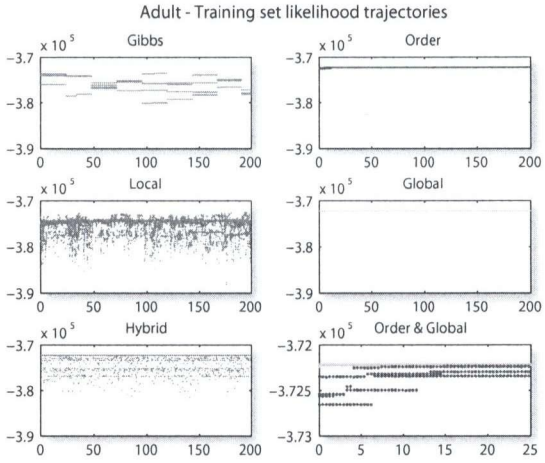


Figure 2.11: **Samplers' training set log likelihood trace plots on Adult dataset.** 4 traceplots of training set likelihood for each sampler, starting from randomly initialized values. The bottom-right figure combines runs from the order and global samplers and shows the behaviour of the chains in the first 25 seconds.

2.5.4 Convergence diagnostics

In Figure 2.11 we show a traceplot of the training set marginal likelihood of the different methods on the Adult dataset. (Other datasets give similar results.) We see that Gibbs is "sticky", that the local proposal explores a lot of poor configurations, but that both the global and order sampler do well. In the bottom right we zoom in on the plots to illustrate that the global sampler is lower variance and higher quality than the order sampler. Although the difference does not seem that large, the other results in this section suggest that the DP proposal does in fact outperform the order sampler.

2.6 Summary and future work

We have proposed a simple method for improving the convergence speed of MCMC samplers in the space of DAG models. Alternatively, our method may be seen as a way of overcoming some of the limitations of the DP algorithm of Koivisto and Sood [31, 32].

The logical next step is to attempt to scale the method beyond its current limit of 22 nodes, imposed by the exponential time and space complexity of the underlying DP algorithm. One way forward might be to sample partitions (layers) of the variables in a similar fashion to [37], but using our DP-based sampler rather than Gibbs sampling to explore the resulting partitioned spaces. Not only has the DP-based sampler been demonstrated to outperform Gibbs, but it is able to exploit layering very efficiently. In particular, if there are d nodes, but the largest layer only has size m, then the DP algorithm only takes $O(d2^m)$ time. Using this trick, [32] was able to use DP to compute exact edge feature posteriors for $d = 100$ nodes (using a manual partition). In future work, we will try to simultaneously sample partitions and graphs given partitions. This is a non-trivial task because the DP algorithm marginalizes over structure. The method of [37], for example, requires DAG samples to estimate parameters associated with partitioning.

Chapter 3

Structure learning with uncertain interventions

3.1 Introduction

The use of Bayesian networks to represent causal models has become increasingly popular [45, 51]. In particular, there is much interest in learning the structure of these models from data. Given observational data, it is only possible to identify the structure up to Markov equivalence. For example, the three models $X \rightarrow Y \rightarrow Z$, $X \leftarrow Y \leftarrow Z$, and $X \leftarrow Y \rightarrow Z$ all encode the same conditional independency statement, $X \perp Z|Y$. To distinguish between such models, we need interventional (experimental) data [16].

Most previous work has focused on the case of "perfect" interventions, in which it is assumed that an intervention sets a single variable to a specific state (as in a randomized experiment). This is the basis of Pearl's "do-calculus" (as in the verb "to do") [45]. A perfect intervention essentially "cuts off" the influence of the parents to the intervened node, and can be modeled as a structural change by performing "graph surgery" (removing incoming edges from the intervened node). Although some real-world interventions can be modeled in this way (such as gene knockouts), most interventions are not so precise in their effects.

One possible relaxation of this model is to assume that interventions are "stochastic", meaning that they induce a distribution over states rather than a specific state [33]. A further relaxation is to assume that the effect of an intervention does not render the node independent of its parents, but simply

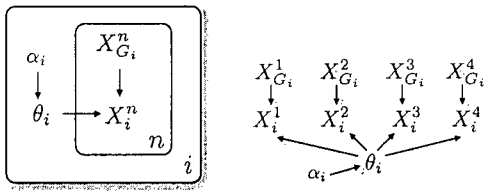
changes the parameters of the local distribution; this has been called a "mechanism change" [52, 53] or "parametric change" [17]. For many situations, this is a more realistic model than perfect interventions, since it is often impossible to force variables into specific states.

Here, we propose a further relaxation of the notion of intervention, and consider the case where the targets of intervention are uncertain. This extension is motivated by problems in systems biology and drug target discovery, where the effects of various chemicals that are added are not precisely known. In particular, each chemical may affect a hidden variable, which can in turn affect multiple observed variables, often in unknown ways. We model this by adding the intervention nodes to the graph, and then performing structure learning in this extended, two-layered graph.

Our contributions are four fold. First, we show how to combine models of intervention — perfect, imperfect and uncertain — with a recently proposed algorithm for efficiently determining the exact posterior probabilities of the edges in a graph [31, 32]. Second, we show empirically that it is possible to infer the true causal graph structure, even when the targets of interventions are uncertain, provided the interventions are able to affect enough nodes. Third, we apply our exact methodology to T-cell data that had previously been analyzed using MCMC [19, 48] and show that our uncertain intervention model is the best density estimator. Fourth, we utilize uncertain interventions to identify gene targets of cancer on the childhood acute lymphoblastic leukemia (ALL) data gathered by [58] and analyzed in [12, 58]. We believe our method is the first well-principled application of Bayesian networks to drug/disease target discovery.

3.2 Models of intervention

We will first describe our probability model under the assumption that there are no interventions. Then we will describe ways to model the many kinds of interventions that have been proposed in the literature, culminating in our



(a) (b)

Figure 3.1: **Intervention model: None.** (a) Plate notation, (b) the same model unrolled across 4 data cases.

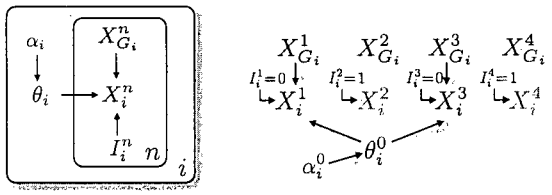
model of uncertain interventions. This will serve to situate our model in the context of previous work.

3.2.1 No interventions

For the intervention-free case, we will assume that the conditional probability distribution (CPD) of each node in the graph is given by $p(X_i|X_{G_i}, \theta, G) = f_i(X_i|X_{G_i}, \theta_i)$, where G_i are the parents of i in G, θ_i are i's parameters, and $f_i()$ is some probability density function (e.g., multinomial or linear Gaussian). For the parameter prior $p(\theta|G)$, we will make the usual assumptions of global and local independence, and parameter modularity (see [27] for details). We will further assume that each $p(\theta_i)$ is conjugate to f_i, which allows for closed form computation of the marginal likelihood $p(X^{1:N}|G) = \int p(X^{1:N}|G,\theta)p(\theta)d\theta$, where N is the number of data cases. For example, for multinomial-Dirichlet, the marginal likelihood for a family (a node and its parents) is given by [27]

$$\begin{aligned} p(x_i^{1:N}|x_{G_i}^{1:N}) &= \int [\prod_{n=1}^{N} p(x_i^n|x_{G_i}^n, \theta_i)]p(\theta_i)d\theta_i \\ &= \prod_{j=1}^{r_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \prod_{k=1}^{q_i} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})} \end{aligned}$$

where $N_{ijk} = \sum_{n=1}^{N} I(x_i^n = k, x_{G_i}^n = j)$ are the counts, and $N_{ij} = \sum_k N_{ijk}$. ($I(e)$ is the indicator function in which $I(e) = 1$ if event e is true and $I(e) = 0$ otherwise.) Also, α_{ijk} are the pseudo counts (Dirichlet hyper parameters),



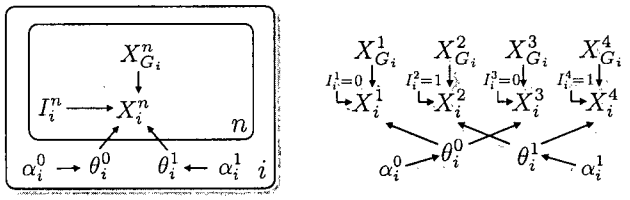
(a) (b)

Figure 3.2: **Intervention model: Perfect.** (a) Plate notation, (b) the same model unrolled across 4 data cases.

$\alpha_{ij} = \sum_k \alpha_{ijk}$, r_i is the number of discrete states for X_i, and q_i is the number of states for X_{G_i}. We will usually use the BDeu prior $\alpha_{ijk} = 1/q_i r_i$ [27]. (An analogous formula can be derived for the normal-Gamma case [23].) The marginal likelihood of all the nodes is then given by $p(X^{1:N}|G) = \prod_{i=1}^{d} p(X_i^{1:N}|X_{G_i}^{1:N})$, where d is the number of nodes. Figure 3.1 shows the non-interventional case as a graphical model.

3.2.2 Perfect interventions

If we perform a perfect intervention on node i in data case n, then we set $X_i^n = x_i^*$, where x_i^* is the desired "target state" for node i (assumed to be fixed and known). We modify the CPD for this case to be $p(X_i|X_{G_i}, \theta) = I(X_i = x_i^*)$. We see that X_i is effectively "cut off" from its parents X_{G_i}. Figure 3.2.(a) shows the perfect intervention model in plate notation, while (b) illustrates the idea on a local family with 4 data points. Namely, for i fixed, Figure 3.2.(b) "unrolls" the plate notation across 4 data. We see that in data cases 2 and 4 (marked in red) the perfect intervention was performed ($I_i = 1$), cutting off X_i from its parents and corresponding parameters. Although not shown, the probability function over X_i's states has been collapsed onto the target state x_i^*.



(a) (b)

Figure 3.3: **Intervention model: Imperfect.** (a) Plate notation, (b) the same model unrolled across 4 data cases. X^n_i is node i in case n, $X^n_{G_i}$ are its parents. I^n_i acts like a switching variable: If $I^n_i = 1$ (representing an intervention), then X_i uses the parameters θ^1_i; If $I^n_i = 0$, then X_i uses the parameters θ^0_i. $\alpha^{0/1}_i$ are the hyper-parameters.

3.2.3 Imperfect interventions

A simple way to model interventions is to introduce intervention nodes, that act like "switching parents": if $I^n_i = 1$, then we have performed an intervention on node i in case n and we use a different set of parameters than if $I^n_i = 0$, when we use the "normal" parameters. Specifically, we set $p(X_i|X_{G_i}, I_i = 0, \theta, G) = f_i(X_i|X_{G_i}, \theta^0_i)$ and $p(X_i|X_{G_i}, I_i = 1, \theta, G) = f_i(X_i|X_{G_i}, \theta^1_i)$. (Note that the assumption that the functional form f_i does not change is made without loss of generality, since θ_i can encode within it the specific type of function.) Tian and Pearl [52, 53] refer to this as a "mechanism change": see Figure 3.3. A special case of this is a perfect intervention, in which $p(X_i|X_{G_i}, I_i = 1, \theta, G) = I(X_i = x^*_i)$. To simplify notation, we assume every node has its own intervention node; if a node i is not intervenable, we simply clamp $I^n_i = 0$ for all n.

When we have interventional data, we modify the local marginal likelihood formula by partitioning the data into those cases in which X_i was passively

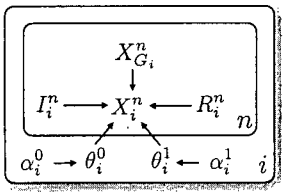


Figure 3.4: **Intervention model: Imperfect with unreliable extension.** Compare to Figure 3.3. We can optionally add another switch node R^n_i, which can be used to model the degree of effectiveness of the intervention.

observed, and those in which X_i was set by intervention:

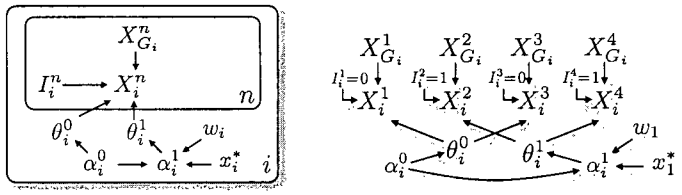
$$\begin{aligned} p(x_i^{1:N}|x_{G_i}^{1:N}, I_i^{1:N}) &= \int [\prod_{n:I_i^n=0} p(x_i^n|x_{G_i}, \theta_i^0)] p(\theta_i^0) d\theta_i^0 \\ &\times \int [\prod_{n:I_i^n=1} p(x_i^n|x_{G_i}, \theta_i^1)] p(\theta_i^1) d\theta_i^1 \end{aligned}$$

In the case of perfect interventions, this second factor evaluates to 1, so we can simply drop cases in which node i was set by intervention from the computation of the marginal likelihood of that node [8].

3.2.4 Unreliable interventions

An orthogonal issue to whether the intervention is perfect or imperfect is the reliability of the intervention, i.e., how often does the intervention succeed? One way to model this is to assume that each attempted intervention succeeds with probability ϕ_i and fails with probability $1 - \phi_i$; this is what Korb et al. [33] call the degree of "effectiveness" of the intervention. We can associate a latent binary variable R_i^n to represent whether or not the intervention succeedeed or failed in case n, resulting in the mixture model

$$\begin{aligned} &p(X_i|X_{G_i}, I_i = 1, \theta, G) \\ &\quad = \sum_r p(R_i = r) p(X_i|X_{G_i}, I_i = 1, R_i = r, \theta, G) \\ &\quad = \phi_i f_i(X_i|X_{G_i}, \theta_i^1) + (1 - \phi_i) f_i(X_i|X_{G_i}, \theta_i^0). \end{aligned} \tag{3.1}$$



(a) (b)

Figure 3.5: **Intervention model: Soft.** (a) Plate notation, (b) the same model unrolled across 4 data cases. Proposed by [38]. x^*_i is the known state into which we wish to force node i when we perform an intervention on it; w_i is the strength of this intervention. α^1_i, the hyper-parameters of θ^1_i, are a deterministic function of α^0_i, x^*_i and w_i.

Figure 3.4 illustrates the idea on the unreliable intervention model. An unreliable, but otherwise perfect, intervention is modeled by setting

$$p(X_i|X_{G_i}, I_i = 1, R_i = 1, \theta, G) = I(X_i = x^*_i).$$

Unfortunately, computing the exact marginal likelihood of a data case now becomes exponential in the number of R variables, because we have to sum over all $2^{|R|}$ latent assignments. Although Figure 3.4 adds the indicator R^n_i to the imperfect model only, any of the other models of intervention under discussion could be augmented with the unreliable assumption also.

3.2.5 Soft interventions

Another way to model imperfect interventions is as "soft" interventions, in which an intervention just increases the likelihood that a node enters its target state x^*_i. Markowetz et al. [38] suggest using the same model of $p(X_i|X_{G_i}, I_i, \theta, G)$ as before, but now the parameters θ^0_i and θ^1_i have *dependent* hyper-parameters. In particular, for the multinomial-Dirichlet case, $\theta^{0/1}_{ij\cdot} \sim Dir(\alpha^{0/1}_{ij\cdot})$, they assume the deterministic relation $\alpha^1_{ij\cdot} = \alpha^0_{ij\cdot} + w_i\vec{e}_t$, where j indexes states (conditioning cases) of x_{G_i}, $t = x^*_i$ is the target value for node i, $\vec{e}_t = (0, \ldots, 0, 1, 0, \ldots, 0)$

with a 1 in the t'th position, and w_i is the strength of the intervention. As $w_i \to \infty$, this becomes a perfect intervention, while if $w_i = 0$ it reduces to an imperfect intervention. If the intervention strength w_i is unknown, Markowetz et al. suggest putting a mixture model on w_i, but it may be more appropriate to use the R_i mixture model mentioned above, where an intervention can succeed or fail on a case by case basis. Figure 3.5 shows the model graphically using plate notation.

3.2.6 Uncertain interventions

Finally we come to our proposed model for representing interventions with uncertain targets, as well as uncertain effects. We no longer assume a one to one correspondence between intervention nodes I_i and "regular" nodes X_i. Instead, we assume that each intervention node I_i may have multiple regular children. (Such interventions are sometimes said to be due to a "fat hand", which "touches" many variables at once.) If a regular node has multiple intervention parents, we create a new parameter vector for each possible combination of intervention parents: see Figure 3.6 for an example.

We are interested in learning the connections from the intervention nodes to the regular nodes, as well as between the regular nodes. We do not allow connections between the intervention nodes, or from the regular nodes back to the intervention nodes, since we assume the intervention nodes are exogenous and fixed. We enforce these constraints by using a two layered graph structure, $V = \mathcal{X} \cup \mathcal{I}$, where $\mathcal{X}$ are the regular nodes and $\mathcal{I}$ are the intervention nodes. The addition of $\mathcal{I}$ motivates new notation, since the augmented adjacency matrix has a special block structure. The full adjacency matrix, denoted by $\mathbf{H}$, is comprised of the *intervention* block $\mathbf{F}$ containing $\mathcal{I}$ nodes, and the *backbone* block $\mathbf{G}$ comprised of $\mathcal{X}$ nodes:

$$\mathbf{H} = \begin{pmatrix} 0 & \mathbf{G} \\ 0 & \mathbf{F} \end{pmatrix}.$$

We call the elements of $\mathbf{F}$ "target edges" since they correspond to edges $\mathcal{I} \to \mathcal{X}$

and the elements of $\mathbf{G}$ "backbone edges". As we will see in Section 3.3.1, the block structure of $\mathbf{H}$ reduces the time complexity of the DP algorithm that we use to perform exact Bayesian inference.

To explain how we modify the marginal likelihood function, we need some more notation. Let X_{G_i} be the regular parents of node i, and I_{G_i} be the intervention parents. Let θ_i^ℓ be the parameters for node i given that its intervention parents have state ℓ. Then the marginal likelihood for a family becomes

$$\begin{aligned} & p(x_i^{1:N} | x_{G_i}^{1:N}, I_{G_i}^{1:N}) \\ &= \prod_\ell \int \left[\prod_{n: I_{G_i}^n = \ell} p(x_i^n | x_{G_i}^n, \theta_i^\ell) \right] p(\theta_i^\ell) d\theta_i^\ell. \end{aligned}$$

It is crucial that we assume that the interventions have local (albeit unknown) effects, otherwise they would not help us resolve Markov equivalency. To see this, note that if the distribution after an intervention, call it $p(X|\theta^1)$, is unrelated to the distribution before an intervention, $p(X|\theta^0)$, then the overall marginal likelihood becomes a product of standard marginal likelihoods (gotten by integrating out θ^0 and θ^1). This gives us more data, but does not help us learn the causal structure (see [53] for more details).

To see this, let us consider a simpler scenario (inspired by the analysis of [53]) in which we have a single intervention node I_i with target $ch(I_i) = \ell$. Suppose we observe N_0 cases in which $I_i = 0$ and N_1 cases in which $I_i = 1$. Let N_{ijk}^0 be the counts in the first batch, N_{ijk}^1 be the counts in the second batch, and $N_{ijk} = N_{ijk}^0 + N_{ijk}^1$. Let G_X be the graph induced by the regular nodes. If the post interventional distribution is unconstrained (i.e., the parameters that generated the second batch of data are unrelated to the first set of parameters), then we get

$$\begin{aligned} & p(X_{1:N_1}, X_{N_1+1:N_2} | I_i^{1:N_1} = 0, I_i^{N_1+1:N_2} = 1, G_X) \\ &= \prod_i \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij}^0)} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk}^0)}{\Gamma(\alpha_{ijk})} \\ &\times \prod_i \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij}^1)} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk}^1)}{\Gamma(\alpha_{ijk})}, \end{aligned}$$

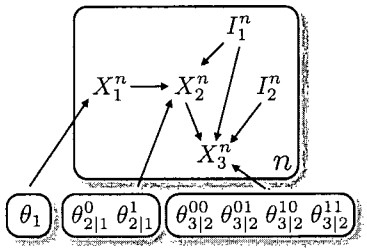


Figure 3.6: **Example of a "Fat hand" intervention.** Intervention 1 affects nodes 2 and 3, intervention 2 affects node 3. The parameters for node 3 are $\theta^{ij}_{3|2}(k,\ell)$, where $I_1 = i$, $I_2 = j$, $X_2 = k$ and $X_3 = \ell$.

which is just the regular BDe likelihood applied to a larger dataset. But if we constrain the intervention to only affect node ℓ, we get

$$\begin{aligned} p(X_{1:N_1}, X_{N_1+1:N_2} | I_i^{1:N_1} = 0, I_i^{N_1+1:N_2} = 1, G_X, \ell) \\ = \prod_{i \neq \ell} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \\ \times \prod_{j=1}^{q_\ell} \frac{\Gamma(\alpha_{\ell,j})}{\Gamma(\alpha_{\ell,j} + N^0_{\ell,j})} \prod_{k=1}^{r_\ell} \frac{\Gamma(\alpha_{\ell,j,k} + N^0_{\ell,j,k})}{\Gamma(\alpha_{\ell,j,k})} \\ \times \prod_{j=1}^{q_\ell} \frac{\Gamma(\alpha_{\ell,j})}{\Gamma(\alpha_{\ell,j} + N^1_{\ell,j})} \prod_{k=1}^{r_\ell} \frac{\Gamma(\alpha_{\ell,j,k} + N^1_{\ell,j,k})}{\Gamma(\alpha_{\ell,j,k})}. \end{aligned}$$

The unconditional marginal likelihood is then given by the mixture distribution

$$p(X|I, G_X) = \sum_\ell p(ch(I_i) = \ell) p(X|I, G_X, \ell).$$

In Section 3.3 we present an efficient way to compute this mixture, even in the case where there are multiple intervention nodes, each with potentially multiple targets.

3.2.7 The power of interventions

The ability to recover the true causal structure (assuming no latent variables) using perfect and imperfect interventions has already been demonstrated both

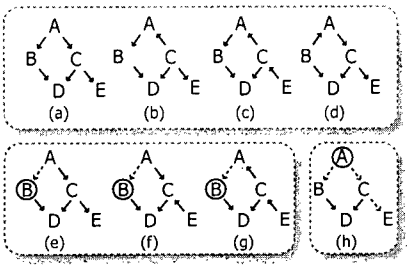


Figure 3.7: **Markov equivalence classes on the Cancer network.** (a) The Cancer network, from [22]. (a-d) are Markov equivalent. (c-g) are equivalent under an intervention on B. (h) is the unique member under an intervention on A. Based on [53].

theoretically [16, 17, 52, 53] and empirically [8, 39, 52, 53, 57]. Specifically, each intervention determines the direction of the edges between the intervened nodes and its neighbors; this in turn may result in the direction of other edges being "compelled" [4].

For example, in Figure 3.7, we see that there are 4 graphs that are Markov equivalent to the true structure; given observational data alone, this is all we can infer. However, given enough interventions (perfect or imperfect) on B, we can eliminate the fourth graph (d), since it has the wrong parents for B. Given enough interventions on A, we can uniquely identify the graph, since we can identify the arcs out of A by intervention, the arcs into D since it is a v-structure, and the $C \rightarrow E$ arc since it is compelled. In general, given a set of interventions and observational data, we can identify a graph up to intervention equivalence (see [52] for a precise definition).

In Section 3.4.1, we will experimentally study the question of whether one can still learn the true structure from uncertain interventions (i.e., when the targets of intervention are a priori unknown), and if so, how much more data one needs compared to the case where the intervention targets are known.

3.3 Algorithms for structure learning

The Bayesian approach to structure learning avoids many of the conceptual problems that arise when trying to combine the results of potentially inconsistent conditional independency tests performed on different ("mutated") models [15]. In addition, it is particularly appropriate when the sample sizes are small, but "soft" prior knowledge is available, as in many systems biology experiments. However, for large structure learning problems Bayesian inference becomes intractable, and we will still have to resort to approximate and/or point estimation methods.

3.3.1 Exact algorithms for $p(\mathbf{H}_{ij})$ and $\mathbf{H}^*_{MAP}$

On problems of less than $d \approxeq 22$ variables, we use the algorithm of Koivisto and Sood [31, 32], which can compute the exact posterior probabilities of all edges, $p(\mathbf{H}_{ij})$, using dynamic programming in $O(d2^d)$ time, as we discussed in Section 2.2. The inputs to this algorithm are a "prior" over node orderings $q_i(U_i)$, a "prior" over possible parent sets, $\rho_i(G_i)$, and a local marginal likelihood function for every node and every possible parent set, $p(X_i|X_{G_i})$. They were also described in Chapter 2.

Though we increase the effective number of nodes in using the uncertain intervention model, the block structure of $\mathbf{H}$ permits a reduction in the time complexity of the DP algorithm. Let $d_I = |\mathcal{I}|$ be the number of intervention nodes, and $d_X = |\mathcal{X}|$ be the number of regular nodes. The time complexity of the DP algorithm in this case is $O(d2^{d_X} + d^{k+1}C(N))$, where $d = d_I + d_X$, and $C(N)$ is the cost of computing each local marginal likelihood term. Note that layering is crucial for efficiently handling uncertain interventions, otherwise the algorithm would take $O(d2^d)$ instead of $O(d2^{d_X})$ time.

Silander and Myllmaki recently devised an elegantly simple algorithm to compute the globally optimal DAG [50], which takes $o(d^2 2^{d-2})$ time and exponential space. The premise is that the optimal DAG must have a sink (a node without outgoing edges), which, by optimality, must have parents that score

the highest amongst all possible sets thereof. If this best sink and its incoming edges are fixed and removed from further consideration, the remaining nodes and edges must be optimal in the same fashion. When carried out recursively, these steps will construct the globally optimal DAG. We use this algorithm to compute the exact MAP, $\mathbf{H}^{*}_{MAP}$.

3.3.2 Local search for $\hat{\mathbf{H}}_{MAP}$

For much larger problems, Bayesian model averaging becomes infeasible to carry out, even by approximation methods. For the 271 variable ALL data (see Section 3.4.3), we will approximate the MAP by finding a structure $\hat{\mathbf{H}}_{MAP}$ by local search that fits the training data well. As with BMA, there also exist two flavours of local search, which differ by the space they operate on: structure or order. In structure search, given a starting DAG, the algorithm considers all neighbouring DAGs that differ by a single edge addition, deletion or reversal then greedily chooses the one that most increases the training set marginal likelihood [27]. Recently, Schmidt et al. [49] compared the two approaches on several datasets and found that DAG search consistently outperformed order search on the structure learning task. Neighbour pruning is an optional preprocessing step that tests for independencies between variables, and rules out many structures or orders a priori. [49] found that if an appropriate neighbour pruning method is applied, DAG search scores better on training and test set likelihood as well. Therefore, we adopt the structure-space local search, forcing it to restart whenever a local maximum is reached, and taking only the highest scoring DAG based on training set marginal likelihood.

3.3.3 Iterative algorithm for $\hat{\mathbf{H}}_{MAP}$

The block-$\mathbf{H}$ notation introduced in Section 3.2.6 is suggestive of a coordinate ascent algorithm, which learns an $\hat{\mathbf{H}}_{MAP}$ by alternating between learning $\mathbf{F}$ given $\mathbf{G}$ fixed, then $\mathbf{G}$ given $\mathbf{F}$ fixed. Even if the number of nodes d_X is large, with a target edge fan-in constraint it will still be feasible to determine $\mathbf{F}_{MAP}$.

This algorithm would seem particularly attractive for applications where the goal is to learn $\mathbf{F}$ only ($\mathbf{G}$ may be irrelevant). We implemented such an algorithm, and tested it according to the same procedures used in Section 3.4.1. However, we found that it becomes consistently trapped in local maxima, especially when the globally optimal DAG algorithm of [50] is used for both steps. The culprit is in fixing $\mathbf{F}$; when we do so, we effectively break the interventional nature of the data and create new local maxima. If the algorithm was allowed to look a sufficient number of steps into the future, it could escape these maxima; however, the extra computation required for the look-ahead would defeat the algorithm's original purpose. The problem can be somewhat ameliorated by using stochastic local search rather than finding the MAP, though in practise it appears to be preferable to simply learn $\mathbf{F}$ and $\mathbf{G}$ jointly.

3.4 Experimental results

We first present some results on synthetic data generated from a Bayesian network of known structure, and then present results on a real biological dataset.

3.4.1 Synthetic data

In this section, we experimentally study the question of whether one can still learn the true structure, even when the targets of intervention are a priori unknown, and if so, how much more data one needs compared to the case where the intervention targets are known[5]. We assessed this using the following experimental protocol. We considered the Cancer network shown in Figures 2.2 and 3.7, and then generated random multinomial CPDs by sampling from a Dirichlet distribution with hyper-parameters chosen by the method described in [6] (outlined in Section 2.5.2). For simplicity, we used binary nodes. We then

[5] Tian and Pearl [52] briefly mention the case of "unknown focal variables" (which we are calling uncertain targets of intervention) in the context of constraint based learning methods, but do not present any algorithms for identifying focal variables. We are not aware of any other papers that address this question.

generated data using forwards sampling; the first 2000 cases D_0 were from the original model, the second 2000 cases D_1 from a "mutated" model, in which we performed a perfect intervention either on A or B, forcing it to the "off" state in each case.

Next we tried to learn back the structure using varying sample sizes of $N \in \{100, 500, 2000\}$. Specifically we used N observational samples and N interventional samples, $D = (D_0^{1:N}, D_1^{1:N})$. We ran the algorithm using data D and under increasingly vague prior knowledge: (1) using the perfect interventions model; (2) using the soft interventions model[6]; (3) using the imperfect model; and (4) using the uncertain interventions model. In the latter case, we also learned the children of the intervention node. As a control, we also tried just using observational data, $D = D_0^{1:2N}$.

Our results for the perfect and uncertain models are shown in Figure 3.8. On this network, the imperfect and soft intervention models perform very similar to the perfect case, though they require more data to achieve the same result. We see that with observational data alone, we are only able to recover the v-structure $B \rightarrow D \leftarrow C$, with the directions of the other arcs being uncertain (e.g., $P(C \rightarrow E) \approx 0.75$.) With perfect interventions on B, we can additionally recover the $A \rightarrow B$ arc, and with perfect interventions on A, we can recover the graph uniquely, consistent with the theoretical results in Section 3.2.7. With uncertain interventions, we see that the entropy of the posterior on the regular edges is higher than when using perfect interventions, but it too reduces with sample size. Eventually the posterior converges to a delta function on the intervention equivalence class. We obtain similar results with other experiments on random graphs. This suggests that our proposed mechanism is able to learn causal structure even from uncertain interventions.

Next, we turned our attention to the larger synthetic network "Cars Diagnosis" introduced by [26], shown in Figure 3.9. Here, we will contrast the structural recovery abilities of the perfect and uncertain models Receiver Operating

[6] [38] do not discuss how to set the pushing strength w_i. We set it equal to $0.5N$, so that the data does not overwhelm the hyper-parameter α_{ijk}^1.

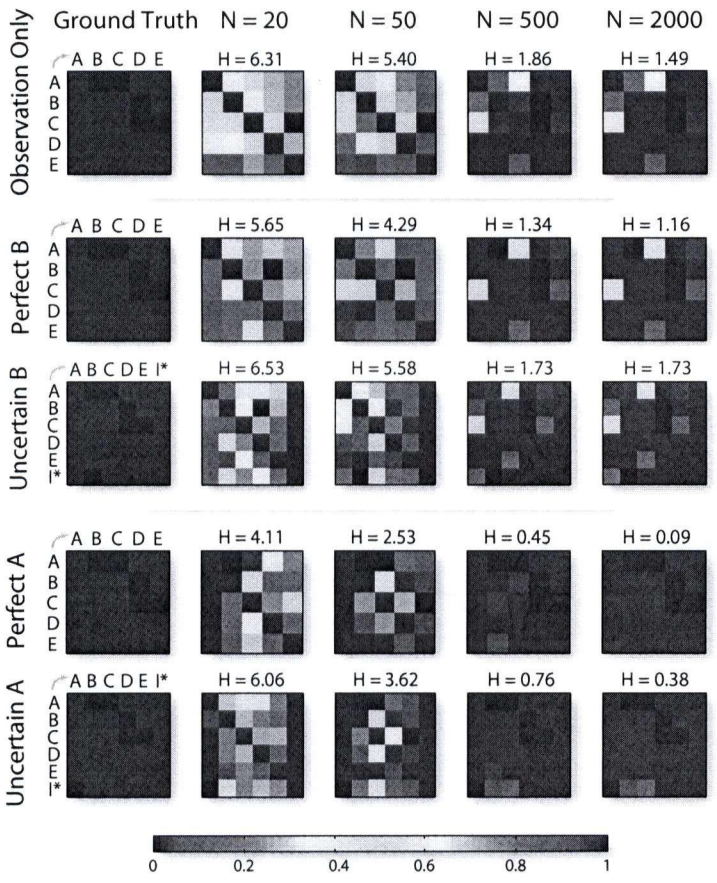


Figure 3.8: **Perfect vs uncertain interventions on the Cancer network.** Results of structure learning on the Cancer network (Figure 3.7). Left column: ground truth. Subsequent columns: posterior edge probabilities $p(G_{ij} = 1|D)$ for increasing sample sizes N, where dark red denotes 1.0 and dark blue denotes 0.0. H is the entropy of the factored posterior $\prod_{ij} p(G_{ij}|D)$. See text for details. This figure is best viewed in colour.

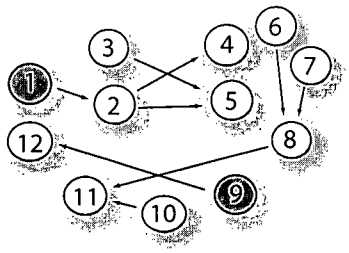


Figure 3.9: **Cars network ground truth.** $d = 12$ model for car malfunction troubleshooting used by [26]. By selecting the appropriate two intervention nodes, marked here in red, it is possible to uniquely recover the structure.

Characteristic (ROC) curve analysis, as in [31]. Again, we assign multinomial CPDs according to the method [6]. Without interventional data, it would be impossible to learn the orientation of some edges on the Cars Diagnosis network; however, if we intervene on nodes 1 and 9, it is possible to uniquely recover the original structure. Referring to Figure 3.9 we can see that under these two interventions, all edges become compelled that were not in the observational case. In this experiment, we account for sampling variability by creating 10 datasets for each of $N = (20, 200, 2000)$. For a given N, half of the data is observational and half interventional, with interventions generated according to the perfect model. Next, we attempted to learn the structure back for all 10 sets of data per sample size N, assuming both the perfect and uncertain models.

Figure 3.10 presents the results using ROC plots to illustrate the effects of sampling different data sets. Since the DP algorithm outputs edge feature marginal probabilities, it is necessary to threshold them to induce hard decisions on edge presence. Let G^{θ}_{ij} denote the marginal (i, j) thresholded by $\theta \in [0, 1]$, and G^{gt}_{ij} be the corresponding ground truth. We say that G^{θ}_{ij} is a *True Positive* (TP) if $G^{\text{gt}}_{ij} = 1 \wedge G^{\theta}_{ij} = 1$, or else if $G^{\text{gt}}_{ij} = 1 \wedge G^{\theta}_{ij} = 0$, then G^{θ}_{ij} is labeled a *False Negative* (FN). Similarly, $G^{\text{gt}}_{ij} = 0 \wedge G^{\theta}_{ij} = 0$ means G^{θ}_{ij} is a *True Negative* (TN), else if $G^{\text{gt}}_{ij} = 0 \wedge G^{\theta}_{ij} = 1$, we say that G^{θ}_{ij} is a *False Positive* (FP). The four classes $\{TP, FP, TN, FN\}$ are mutually exclusive and exhaustive. Next,

let $\#TP$ indicate the sum of true positives over (i, j), with identical definitions for the other classes. We now define $Sensitivity = \#TP/(\#TP + \#FN)$, $Specificity = \#TN/(\#FN+\#TN)$ and $InverseSpecificity = 1-Specificity$. A ROC curve shows the inherent trade-off between specificity and sensitivity as the threshold θ is varied. ROC plots are often summarized by a single quantity known as the AUC, which is simply the area under the curve. A perfect predictor receives an AUC of 1, while a random predictor gets 0.5.

Given that the interventions are perfect in the generating network, we expect the matching assumption for learning to perform best. As we can see from Figure 3.10, this is the case on average but the difference is not significant. As the sample size is increased, the gap between uncertain and perfect intervention performance closes. The AUC scores for the target edges suggest that learning the target edges requires less data, and happens before the backbone is determined.

In Section 3.4.3, we analyze a $d = 271$ dataset for which Bayesian inference is intractable and instead use local search to approximate the MAP, yielding $\hat{\mathbf{H}}_{MAP}$. However, it is not clear how much is lost when estimating structure via approximate plugin (local search), versus exact plugin (MAP), versus exact BMA. Therefore, we repeated the synthetic Cars experiments, but this time comparing BMA, MAP and local search, all under the uncertain intervention model. For sample sizes of $N \in 20, 100, 400, 2000$, we generated 20 independent datasets and ran each algorithm to obtain $\mathbf{H}$ as edge marginal probabilities (BMA) or single-DAG estimates (MAP and local search). Local search was allowed to run for 20 seconds (3 times as long as it took to compute the BMA or MAP). Since ROC curves do not exist for hard assignments, we instead chose to show the specificity and sensitivity of each estimate. For BMA, we selected the equal-error rate point to threshold the marginals. This is the threshold for which the specificity equals the sensitivity, a fair measure of performance for this experiment. The results in Figure 3.11 suggest that except for extremely low sample sizes, point estimates are adequate estimates of both the backbone and target edges. Furthermore, local search performs within one standard deviation

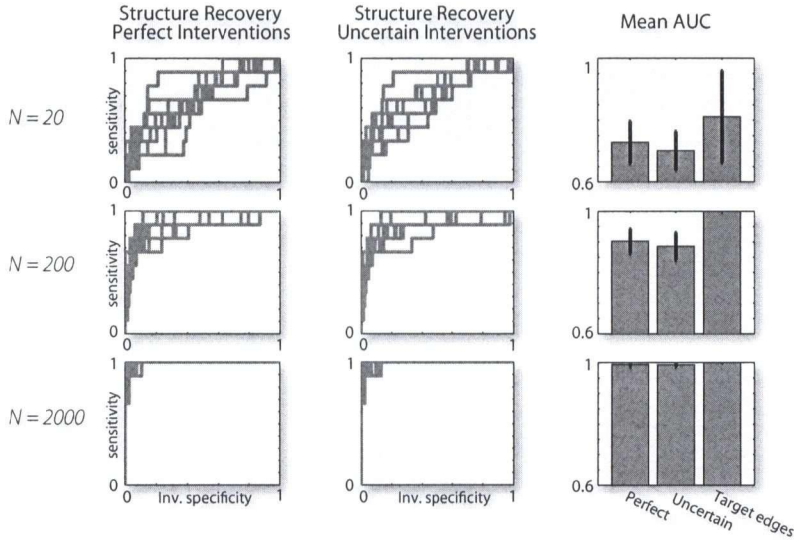


Figure 3.10: **Structural recovery performance of perfect vs uncertain intervention models on Cars network.** Each row of the figure denotes a different sample size ($N = 200$ means there were 100 observational and 100 interventional data cases). The first two columns contain ROC curves showing the trade-off between specificity and sensitivity in thresholding the edge feature marginals produced by perfect and uncertain interventions. There are 10 curves per plot, corresponding to the 10 datasets generated per sample size. In the right column, the ROC curves have been summarized as AUC. The "Perfect" and "Uncertain" bars show structure recovery performance for the graph backbone (**G**), while "Target edges" applies only to the uncertain model, and is the AUC on intervention to backbone edges only (**H**).

of the exact MAP estimate.

3.4.2 T-cell data

We now apply our methodology to a real biological dataset, which had previously been analyzed using MCMC by Sachs et al. [48] (who used multiple restart simulated annealing in the space of DAGs), Werhli et al. [57] (who used Metropolis-Hastings in the space of node orderings), and Ellis and Wong [19] (who used equi-energy sampling in the space of node orderings). The purpose of our experiment is to determine the exact posterior over edges, and hence to assess the quality of the MCMC techniques, and also to learn the effects of the interventions that were performed.

The dataset consists of 11 protein concentration levels measured under 6 different interventions, plus 3 unperturbed measurements. The proteins in question constitute part of the signaling network of human T-cells, and therefore play a vital role in the immune system. See Figure 3.13(a) for a depiction of the commonly accepted "ground truth" network, including hidden nodes.

The data in question were gathered using a technique called flow cytometry, which can record phosphorylation levels of individual cells. This has two advantages compared to other measurement techniques: first, it avoids the information loss commonly incurred by averaging over ensembles of cells; second, it creates relatively large sample sizes (we have $N = 5400$ data points in total, 600 per condition).

The raw data was discretized into 3 states, representing low, medium and high activity. We obtained this discretized data directly from Sachs; see Figure 3.12 for a visualization. This constituted the input to our algorithm.

We tried two different analyses. In the first version, we assumed that the targets of intervention were known, and we modeled these using perfect interventions (as did Sachs et al.). The results are shown in Figure 3.13(c). These should be compared with the results of the MCMC analysis of Sachs et al., which are shown in Figure 3.13(b), and the ground truth network, which is shown in

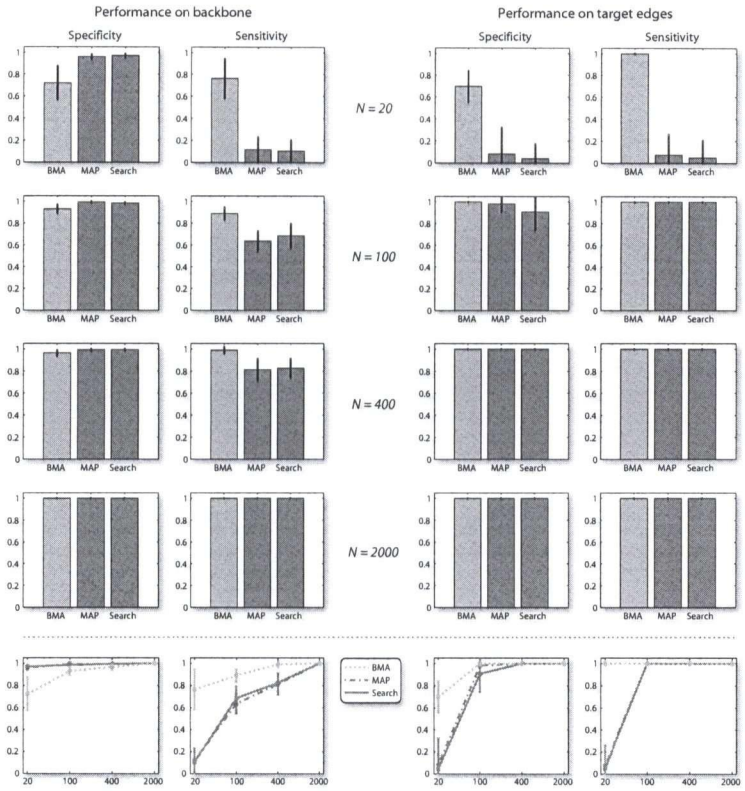


Figure 3.11: **Structural recovery performance of exact BMA vs point estimation on Cars network.** The leftmost two columns show performance on the graph backbone (**G**) while the rightmost two columns show it for the target edges (**F**). The results in each row were aggregated from 20 independent datasets with a sample size of N (shown). $N = 200$ means 100 interventional data and 100 observational data. The bottom row summarizes the plots above as curves. Note that since we chose the equal error rate threshold for BMA, we would expect the associated bar heights to be equal. This is not the case because there is no threshold setting such that $Specificity(\theta) = Sensitivity(\theta)$ exactly. We note that the equal error rate threshold also corresponds to the maximum value taken on by $f(\theta) = Specificity(\theta) + Sensitivity(\theta)$ therefore we take the maximum of $f(\theta)$ as our operating point.

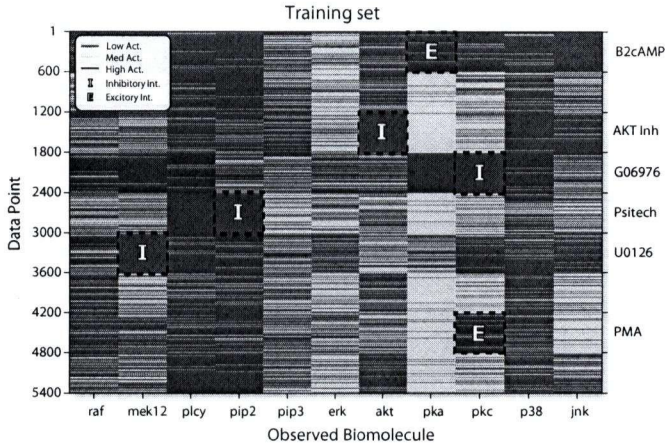


Figure 3.12: **T-cell dataset.** Discretized, 3-state data from [48]. Columns are the 11 measured proteins, rows are the 9 experimental conditions, 3 of which are "general stimulation" rather than specific interventions. The name of the chemical that was added in each case is shown on the right. The intended primary target is indicated by an E (for excitation) or I (for inhibition). This figure is best viewed in colour.

Figure 3.13(a).

While there is substantial agreement between the three models, there are also many differences. For example, the ground truth shows no edge from jnk to p38, or from mek12 to jnk, yet both inference methods detect such an edge. This may be due to the presence of various hidden variables. Looking at the data in Figure 3.12, mek12 and jnk seem quite highly correlated, although this is obviously not enough evidence to suggest there should be an edge between them (as shown in [48], nearly all of the variables are significantly pairwise correlated).

There are also several edges in our model that seem to be absent in the MCMC analysis of Sachs et al. This is possibly because Sachs et al. only perform model averaging over a "compendia of high scoring networks", as found by 500 restarts of simulated annealing, whereas our method averages over all graphs, and hence may detect support for many more edges. (Note that averaging over many sparse, but different, graphs can result in a dense set of marginal edge probabilities.) Also, the two methods use different graph priors $p(G)$, and hence cannot be directly compared.

We also applied the algorithm of Silander and Myllmaki [50] to compute the MAP DAG. On this dataset, it coincides exactly with Figure 3.13(c), suggesting that the posterior is highly peaked around the MAP structure. Since the algorithm does not sum over orderings, the bias discussed in Section 2.3 is not incurred, and the output MAP estimate is with respect to a uniform prior over DAGs.

In the second experiment, we added the intervention nodes to the graph and learned their children, rather than pre-specifying them. The results are shown in Figure 3.13(d). We successfully identified the known targets of all but one of the 6 interventions. (We missed the G06967 $\rightarrow$ pkc edge.) However, we also found that the interventions have multiple children, even though they were designed to target specific proteins. Upon further investigation, we found that each intervention typically affected a node and some of its immediate neighbors. For example, from the ground truth network in Figure 3.13(a), we see that Psitect

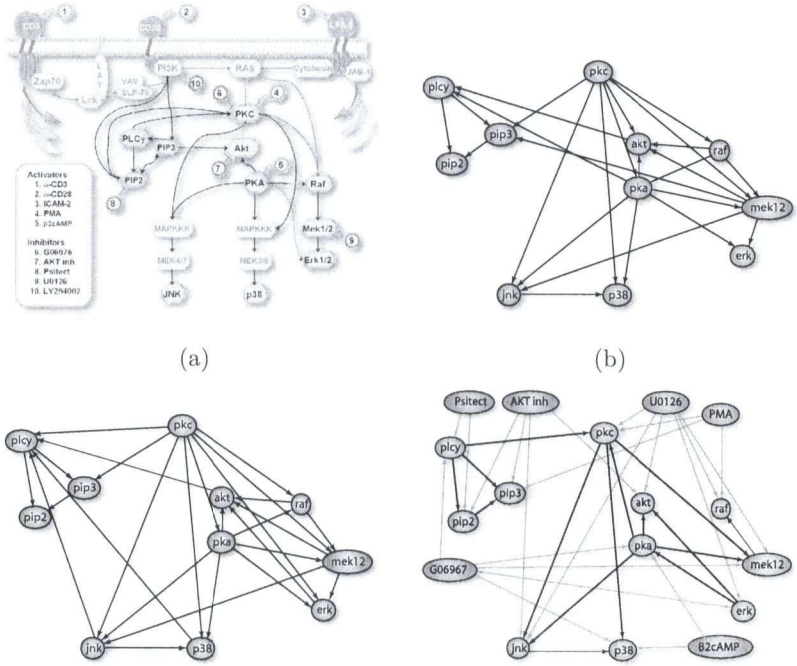
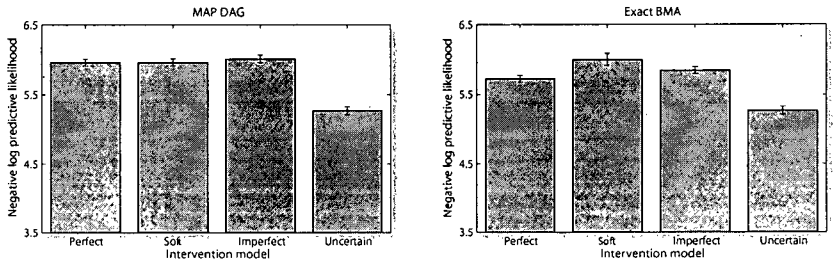


Figure 3.13: **T-cell dataset results.** Models of the biological data. (a) A partial model of the T-cell pathway, as currently accepted by biologists. The small round circles with numbers represent various interventions (green = activators, red = inhibitors). From [48]. Reprinted with permission from AAAS. (b) Edges with marginal probability above 0.5 as estimated by [48]. (c) Edges with marginal probability above 0.5 as estimated by us, assuming known perfect interventions. (d) Edges with marginal probability above 0.5 as estimated by us, assuming uncertain, imperfect interventions, and a fan-in bound of $k = 2$ for the target edges. The intervention nodes are in red, and edges from the intervention nodes are light gray. This figure is best viewed in colour.



(a) (b)

Figure 3.14: **Predictive likelihood scores of intervention models on T-cell dataset.** Negative log predictive likelihood. *Lower is better.* (a) MAP estimate (b) Exact BMA. Result obtained across 10-fold validation. Note: though difficult to see on this scale, exact BMA outperforms the plug-in estimate for uncertain interventions.

(designated 8 in that figure) is known to inhibit pip2; in our learned network (Figure 3.13(d)), we see that Psitect connects to pip2, but also to plcy, which is a neighbor of pip2. This is biologically plausible, since some of these interventions actually work by altering hidden variables, which can therefore cause changes in several neighboring visible variables. Also, although we missed the G06967 $\rightarrow$ pkc edge, the other children of G06967 (plcy, pka, mek12, erk and p38) seem to be strongly affected by G06967 when looking at the data in Figure 3.12. We also computed the MAP DAG and again found it to be identical to the thresholded edge marginals gotten by DP.

We also tried analysing the continuous data using linear-Gaussian Bayes nets [23]. Following [19], we took a log transform of each variable and then standardized them. Our results are similar to [19], but our graph is much denser, suggesting that their MCMC scheme failed to visit sufficiently many modes. (Although once again our results are not directly comparable due to the different prior.) The graphs inferred using the Gaussian and multinomial models have much in common, but they also differ in many of the details.

It is difficult to rigorously assess the quality of the obtained graphs when

there is no ground truth. (The biological model in Figure 3.13(a) is unlikely to be the "true" model that generated the data in Figure 3.12. Also, it contains hidden variables, so is not directly comparable to what we are learning.) The approach taken by Ellis et al. [19] was to compare the predictive log-likelihood in a cross-validation framework. This can also be done using the DP algorithm, by computing $p(x|D) = p(x, D)/p(D)$; these normalization constants can be obtained by running the "forwards" algorithm of [32] using the "dummy" feature $f = 1$. Using 10-fold cross-validation we carried this procedure out for all of the intervention models. We also computed the predictive log-likelihood for the MAP structure under each model. Given the MAP DAG, this amounts to determining its posterior mean parameters, then evaluating the likelihood of each test point in the fold. Our results are shown in Figure 3.14. We see that our uncertain intervention model is the clear victor in both cases, suggesting that the assumption of perfect interventions may be poor for the T-cell data.

Running time on T-cell data

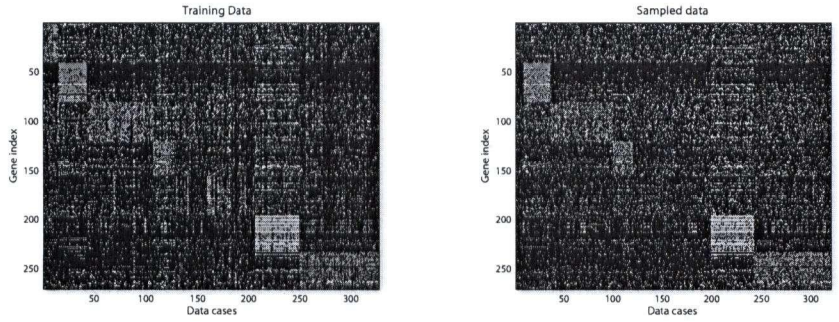
Experiments were performed on a laptop with a 2 GHz Intel Core Duo Processor and 2GB RAM running under Windows XP. For the 3-state T-cell data, with $d = 11$ nodes (using perfect interventions and a fan-in constraint of $k = 5$) and $N = 5400$, our Matlab implementation took 3.6 seconds to compute the marginal likelihood terms, while the DP algorithm took 0.4 seconds and the MAP DAG algorithm needed 0.8 seconds. For the case where we learned the effects of interventions (so $d = 17$), it took about 3.6 minutes (using a fan-in bound of $k = 5$ for backbone edges and $k = 2$ for target edges) to obtain the likelihood terms, 15 seconds for BMA and 80 seconds for the MAP. By comparison, the multiple restart simulated annealing approach used by Sachs et al. took several days.

3.4.3 ALL data

In this section we explore a promising application of the uncertain intervention model: drug/disease target discovery in gene networks. The complex interactions in most gene networks are poorly understood, but there is hope that they can be estimated from measurements such as gene expression microarray data. Learning a protein signalling network under normal cellular conditions was the goal in Section 3.4.2, but a more interesting, if more difficult, application is to determine the effect of external agents like drugs or disease on gene interactions. A biologist may or may not want to learn the backbone network as well.

Since our intervention model directly enables this type of analysis, we apply it to the gene expression data obtained by Yeoh et al. [58], which was previously analyzed in [12, 58]. The data is comprised of measurements of 12,000 genes, from 327 humans suffering from different forms of acute lymphoblastic leukemia (ALL). ALL is a heterogeneous cancer, meaning that it is manifested by several subtypes that vary by their genetic influence and consequently in their response to treatment. The dataset of [58] contains 7 classes, 6 of which represent the common ALL subtypes HYPERDIP > 50, E2A-PBX1, BCR-ABL, TEL-AML1, MLL and T-ALL, and the final class aggregating several less common subtypes. We followed [12] and omitted all but 271 genes from our analysis, using the Chi-square-based filtering method of [58] which selects the top 40 discriminative genes for each subtype (9 genes were chosen multiply chosen across subtypes, yielding 271 *unique* genes). We also discretized the data to 3 levels "underexpressed" (+1), "unchanged" (0) and "overexpressed" (−1) using the same procedure as [12]. Specifically, if (μ_i, σ_i) were the mean and standard deviation of gene i, values less than $\mu_i - \sigma_i$ were mapped to -1, greater than $\mu_i + \sigma_i$ to $+1$ and the remainder to 0. The resulting dataset is shown in Figure 3.15.(a).

Dejori et al. assume that each ALL subtype acts on a single gene, which then causes other genes downstream in the network to change. They learned a Bayesian network using simulated annealing search on the 327-case dataset, ignoring the fact that the data samples come from different conditions. Let

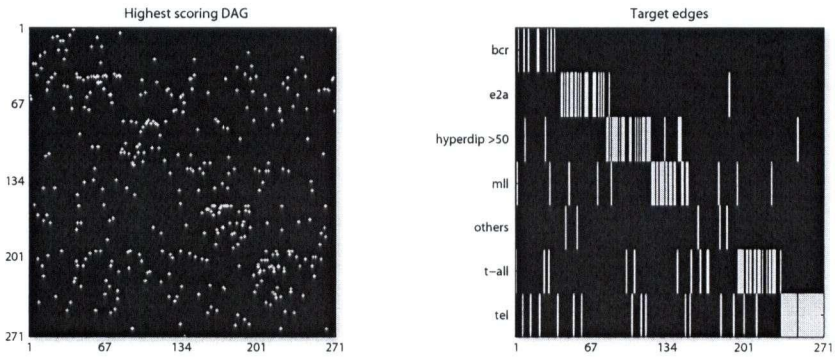


(a) (b)

Figure 3.15: **ALL dataset.** (a) Training dataset (b) data sampled from model under same conditions as training set (cancer subtype orders and frequencies). This figure is best viewed in colour.

D_k denote all data cases representing ALL subtype index k ranging over the 7 subtypes. Using exact inference they generated a sample $D'_i = \{x \sim p(X_{-i}|X_i = +1)\}$ and another $D'_i = \{x \sim p(X_{-i}|X_i = -1)\}$ for each gene i, and compared the Euclidean distance between D'_i and D_k for each k. They declared the cause of type k cancer to be the gene that, when overexpressed, generated data D'_i that most closely resembled D_k. Note that they use a Bayesian network simply as a density estimator for discrete data; they make no attempt to interpret the structure of the graph. When they set $X_i = +1$ or $X_i = -1$, they treat this as an observation, rather than a Pearl-style do-action, so they could have in principle used any other kind of density estimator.

There are two obvious shortcomings to the approach of [12]. First, if we believe that the presence of cancer gives rise to changes in the gene interactions, then it is not sensible to learn a single Bayesian network across multiple cancer conditions. The analysis of Dejori et al. avoids this issue by not discussing the graph structure they learn. However, they do so implicitly when they use their Bayesian network to simulate data from "no cancer" condition. Secondly, from a computational standpoint, their method is very expensive due to the need for



(a) (b)

Figure 3.16: **Highest scoring DAG found on ALL dataset.** Highest scoring found across 20 parallel, 24 hour-long runs of restarting local search. (a) DAG backbone, $\mathbf{G}$, (b) corresponding target edges, $\mathbf{F}$. Next-best structure had a score approximately e^{64} times lower.

inference on a large network to sample from $p(X_{-i}|X_i = \pm 1)$. This problem would be greatly compounded if they considered the cause of ALL to be the mutation of $b \leq B$ genes, since this would require inference to be performed $O(\binom{d}{b})$ times per subtype.

In our approach, we augment the 271 backbone variables with 7 binary intervention nodes encoding the presence or absence of the ALL subtypes and attempt to learn $\mathbf{H}$. Following [12], we assume that each gene has at most one cancer subtype parent. The problem size is well beyond the limitations of the exact DP or MAP algorithms, therefore we use local search, modified to support uncertain interventions. We tried the MMPC neighbour pruning algorithm introduced in [56] as a way improve search, but found that even though unrestricted DAG search is slower by a factor of 4, it finds better-scoring graphs. MMPC likely gave poor results because it identifies sets of potential neighbours (parents and children) using conditional independency tests that are undoubtably unreliable with a sample size of 371 for $d = 271$ variables. We

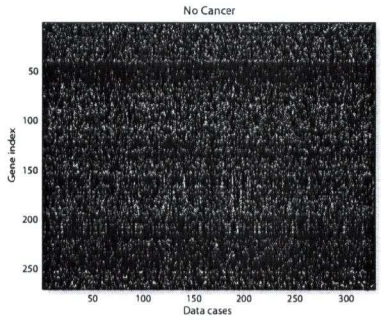


Figure 3.17: **Predicted gene-expression profile for a patient without ALL.** The procedure of [12] is arguably incapable of estimating these profiles. Their Figure 3.(b) appears to support this claim.

ran 20 searches seeded at random initializations for 24 hours each, and allowed them to restart at a new random DAG whenever they became caught in a local maximum. In the results that follow, we adopt the highest scoring DAG $\hat{\mathbf{H}}_{LS}$ as our plugin estimate of the structure, $\mathbf{H}$. Figure 3.16 shows the graph. We note that $\hat{\mathbf{H}}_{LS}$ shared substantial similarity with other high scoring graphs.

Since our model is generative we can sample data from it and then compare this data to the training data to determine if the model is sensible. Figure 3.15.(b) displays data sampled from $\hat{\mathbf{H}}_{LS}$ under the same conditions as the training set, meaning that in the cases where, for example, HYPERDIP > 50 was present in the training data, we clamped the subtype's corresponding intervention node to "on" for the same cases in the synthetic dataset. It is apparent that our model has captured much of the detail from the original data. In Figure 3.17 we sample 327 cases from the *non-interventional* condition. This condition was not contained in the training data, but our model is capable of learning it through the assumed locality of interventions. Figure 3.18.(a)-(b) shows the expression profile for 327 simulated patients with subtype E2A-PBX1 or MLL. Our model can also estimate expression profiles for hypothetical patients who have unluckily contracted more than one subtype of ALL. One such

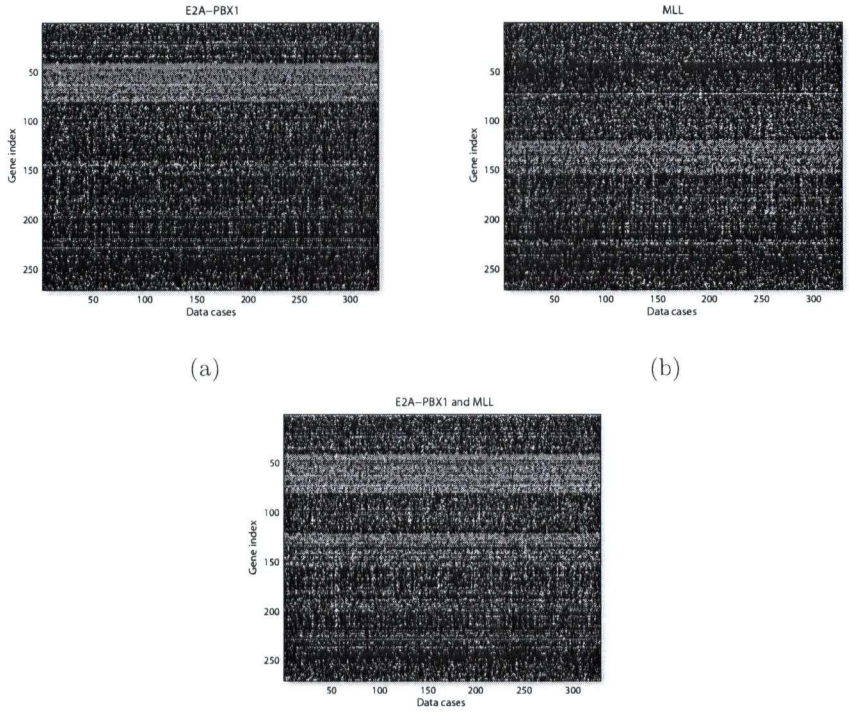


Figure 3.18: **Sampled gene-expression profile for a patient with ALL subtype (a) E2A-PBX1 or (b) MLL or (c) *both*.**

combination, E2A-PBX1 *and* MLL, is shown in Figure 3.18.(c), which appears to be reasonable after referring to (a) and (b).

The main objective of this analysis is to determine the genetic targets of ALL. These are easy enough to read off of $\hat{\mathbf{F}}_{LS}$ and can be seen in Figure 3.16.(b). As can be seen, the model has picked up many targets for each subtype, especially for TEL-AML1. We require a method to rank these edges in order to compare our results with [12], who report the top 5 scoring gene targets per ALL subtype. One method would be to analyze the top M scoring DAGs found by local search, and assign a score to all of the target edges in their union according to the number of times each edge appeared. If the DAGs were samples from the posterior, this would be valid; however, greedy local search is certainly not sampling from the posterior. Instead, we adopt a more principled approach by computing, for each target edge in $\hat{\mathbf{F}}_{LS}$, the cost of removing that edge with all other structure being fixed. The expression for this weight, denoted $W_{i,x}$ with $i \in \mathcal{I}$ an intervention node and $x \in \mathcal{X}$ a backbone gene node, is given by:

$$W_{i,x} = \log p(D|G_x, i \rightarrow x) - \log p(D|G_x), \tag{3.2}$$

where $G_x \in \mathcal{X}$ are the backbone parents of x and $p(D|\cdot)$ are local marginal likelihood terms. $W_{i,x}$ is the plugin estimate to a Bayes factor that answers the same question, but integrates over the remaining structure rather than holding it fixed. We can also compute $W_{i,x}$ on the edges not chosen by $\hat{\mathbf{F}}_{LS}$, with the quantity now representing the gain in adding these edges. By the fact that local search find a local maximum, $W_{i,x}$ is certain to be positive for edges turned "on" in $\hat{\mathbf{F}}_{LS}$ and negative otherwise.

With $W_{i,x}$ we can plot a spectrum of target edge scores across the 271 genes, for each subtype. These are shown in Figures 3.19-3.21.(a). Note that these represent "monogenic activations" only. We could also compute $W_{i,x}$ for vector arguments of i, but we follow [12] and only report results on single gene perturbations. Also, many of the negative $W_{i,x}$ are not shown as their score is negative infinity. These infinite values arise from the initial assumption that genes can only be affected by one cancer. Therefore, sparsity/density on the plot

shows which genes do not have any parent in the intervention (cancer subtype) layer. These figures also show where our results overlap with [12]; genes which Dejori et al. chose as their top 5 are shown in red with a black square instead of a circle. Next to the squares is a number between 1 and 5 that indicates how highly [12] ranked the gene to be a cause of ALL. Referring to Figure 3.19, we see that our top-scored gene for subtypes E2A-PBX1 and BCR-ABL agree with Dejori et al.'s. This is a positive result, because these genes are proto-oncogenes suspected to cause those ALL subtypes. Our respective spectra for subtype E2A-PBX1 also closely agree, though it is difficult to take the comparison any further, since we do not have access to their detailed results.

Another interesting capability of our model is in determining the type of interaction between the cancer and the genes it targets. Given $\hat{\mathbf{F}}_{LS}$ we compute its posterior mean parameters and then examine the conditional probability table for a particular gene target X_i that has a cancer parent I_i. We marginalize the backbone parents, and look at the parameters corresponding to the "on" state of the cancer:

$$p(X_i|I_i = 1) = \frac{\sum_{X_{G_i}} p(X_i|X_{G_i}, I_i = 1)}{\sum_{X_i, X_{G_i}} p(X_i|X_{G_i}, I_i = 1)}.$$

If most of the mass resides in the "overexpressed" state, $p(X_i = +1|I_i = 1)$, we say the cancer is excitory for that gene, while if the "underexpressed" state, $p(X_i = -1|I_i = 1)$, dominated we would say it is inhibitory. We plot the expression of target genes in Figures 3.19-3.21.(b). Red upwards arrows indicate excitation, while blue downwards arrows show inhibition. The remaining probability mass corresponding to the "no change" state is not shown, but can be easily derived from the fact that the three probabilities sum to unity.

3.5 Summary and future work

We have shown how to apply the dynamic programming algorithm of Koivisto and Sood [31, 32] to learn causal structure from interventional data. We then introduced the model of uncertain interventions, which enables the discovery of

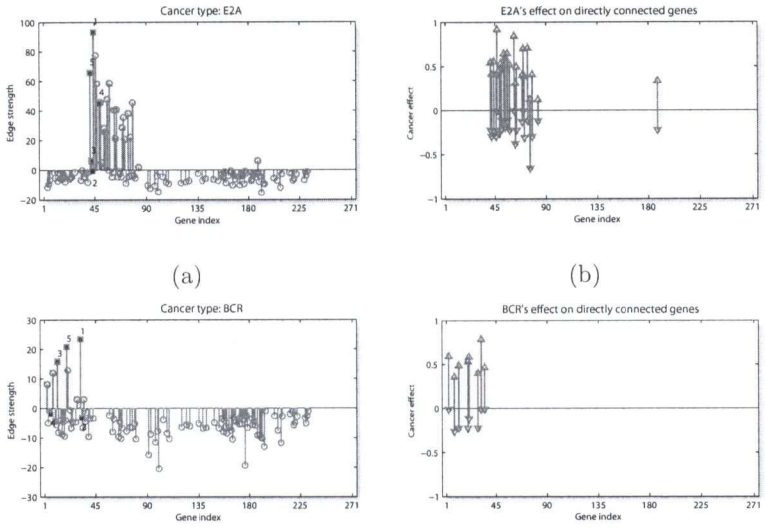


Figure 3.19: **ALL subtypes E2A-PBX1 and BCR-ABL results.** E2A-PBX1: (a)-(b), BCR-ABL: (c)-(d). *Left:* Target edge strength $W_{i,x}$ for chosen edges (positive) and absent edges (negative). Points in red marked by a square indicate overlap with the results of [12]. Gaps correspond to target edges which are impossible. The result for E2A-PBX1 agrees strongly with the corresponding spectrum of [12]. *Right:* Cancer's effect on gene's expression level. This analysis is sensible only for target edges that were chosen (i.e. present in $\hat{\mathbf{F}}_{LS}$).

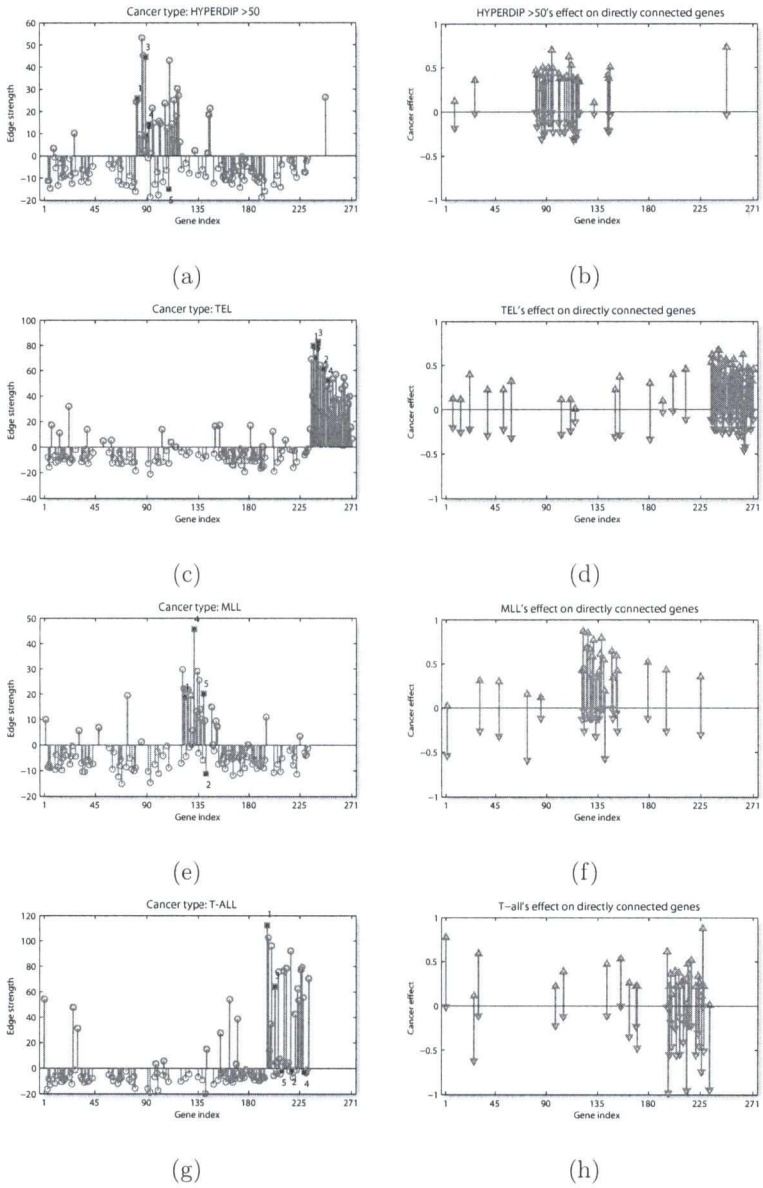
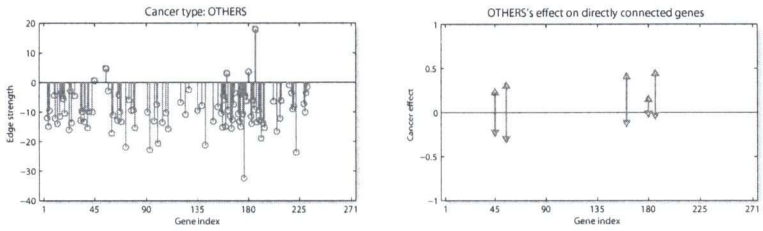


Figure 3.20: **ALL subtypes Hyperdip > 50, TEL-AML1, MLL and T-ALL results.** Hyperdip > 50: (a)-(b), TEL-AML1: (c)-(d), MLL: (e)-(f), T-ALL: (g)-(h). *Left:* Target edge strength $W_{i,x}$ *Right:* cancer's effect on expression level.



(a) (b)

Figure 3.21: **Other ALL subtypes' results.** (a) Target edge strength $W_{i,x}$ (b) cancer's effect on expression level. No corresponding result to (a) is presented in [12], presumably because this class does not represent a homogeneous ALL subtype.

the target of an intervention. Leveraging the dynamic programming algorithm, we established that little data is needed to learn intervention targets, while only modestly more is required to simultaneously determine the graph backbone. We applied this model to two gene expression datasets that are of great relevance to modern systems biology, and demonstrated novel results in both cases.

In the near future, we will apply the uncertain intervention model to the data of Ideker et al. [30], which consists of gene expression data from the galactose pathway of the yeast *Saccharomyces cerevisiae*. Using another methodology, Hallen et al. [25] report learning the gene targets of the compound Galactose; we expect to reproduce this result using our approach.

Another interesting extension would be to apply the uncertain intervention idea to the active learning case [44, 54], where one has to decide which interventions to perform.

Bibliography

[1] A.-L. Barabsi and Z. N. Oltvai. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5:101–113, 2004.

[2] G. Brightwell and P. Winkler. Computing linear extensions is #P-complete. In *STOC*, 1991.

[3] W. Buntine. Theory refinemement on Bayesian networks. In *UAI*, 1991.

[4] D. Chickering. A transformational characterization of equivalent Bayesian network structures. In *UAI*, 1995.

[5] D. Chickering, D. Heckerman, and C. Meek. A Bayesian Approach to Learning Bayesian Networks with Local Structure. In *UAI*, 1997.

[6] D. Chickering and C. Meek. Finding Optimal Bayesian Networks. In *UAI*, 2002.

[7] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory*, 14:462–67, 1968.

[8] G. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *UAI*, 1999.

[9] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[10] D. Dash and G. Cooper. Model Averaging for Prediction with Discrete Bayesian Networks. *J. of Machine Learning Research*, 5:1177–1203, 2004.

[11] N. de Freitas, P. Hjen-Srensen, M. I. Jordan, and S. Russell. Variational MCMC. In *UAI*, 2001.

[12] Mathäus Dejori and Martin Stetter. Identifying interventional and pathogenic mechanisms by generative inverse modeling of gene expression profiles. *Journal of Computational Biology*, 11(6):1135–1148, 2004.

[13] D. Eaton and K. Murphy. Bayesian structure learning using dynamic programming and MCMC. In *UAI*, 2007.

[14] D. Eaton and K. Murphy. Exact Bayesian structure learning from uncertain interventions. In *AI/Statistics*, 2007.

[15] F. Eberhardt. Sufficient condition for pooling data from different distributions. In *First Symposium on Philosophy, History, and Methodology of Error*, 2006.

[16] F. Eberhardt, C. Glymour, and R. Scheines. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among N variables. In *UAI*, 2005.

[17] F. Eberhardt, C. Glymour, and R. Scheines. Interventions and causal inference. In *20th Mtg. Philos. of Sci. Assoc.*, 2006.

[18] D. Edwards. *Introduction to graphical modelling.* Springer, 2000. 2nd edition.

[19] B. Ellis and W. Wong. Sampling Bayesian Networks quickly. In *Interface*, 2006.

[20] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of statistics*, 28(2):337–374, 2000.

[21] N. Friedman and D. Koller. Being Bayesian about Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning*, 50:95–126, 2003.

[22] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *UAI*, 1998.

[23] D. Geiger and D. Heckerman. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *The Annals of Statistics*, 30(5):1412–1440, 2002.

[24] P. Giudici and R. Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50(1–2):127–158, January 2003.

[25] K. Hallen, J. Bjorkegren, and J. Tegner. Detection of compound mode of action by computational integration of whole genome measurements and genetic perturbations. *BMC Bioinformatics*, 7(51), 2006.

[26] D. Heckerman, J. Breese, and K. Rommelse. Troubleshooting under uncertainty. Technical Report MSR-TR-94-07, Microsoft Research, 1994.

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

[28] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19:2271–2282, 2003.

[29] K.-B. Hwang and B.-T. Zhang. Bayesian model averaging of Bayesian network classifiers over multiple node-orders: application to sparse datasets. *IEEE Trans. on Systems, Man and Cybernetics*, 35(6):1302–1310, 2005.

[30] T. Ideker, V. Thorsson, J. Ranish, R. Christmas, J. Buhler, R. Bumgarner, R. Aebersold, and L. Hood. Integrated genomic and proteomic analysis of a systematically perturned metabolic network. *Science*, 2001. Submitted.

[31] M. Koivisto. Advances in exact Bayesian structure discovery in Bayesian networks. In *UAI*, 2006.

[32] M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *J. of Machine Learning Research*, 5:549–573, 2004.

[33] K. Korb, L. Hope, A. Nicholson, and K. Axnick. Varieties of causal intervention. In *Pacific Rim Conference on AI*, 2004.

[34] D. Madigan, J. Gavrin, and A. Raftery. Enhancing the predictive performance of Bayesian graphical models. *Communications in Statistics - Theory and Methods*, 24:2271–2292, 1995.

[35] D. Madigan and A. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *J. of the Am. Stat. Assoc.*, 89:1535–1546, 1994.

[36] D. Madigan and J. York. Bayesian graphical models for discrete data. *Intl. Statistical Review*, 63:215–232, 1995.

[37] V. Mansinghka, C. Kemp, J. Tenenbaum, and T. Griffiths. Structured priors for structure learning. In *UAI*, 2006.

[38] F. Markowetz, S. Grossmann, and R.Spang. Probabilistic soft interventions in Conditional Gaussian networks. In *10th AI/Stats*, 2005.

[39] F. Markowetz and R. Spang. Evaluating the effect of perturbations in reconstructing network topologies. In *Proc. 3rd Intl. Wk. on Distrib. Stat. Computing*, 2003.

[40] M. Meila and T. Jaakkola. Tractable Bayesian learning of tree belief networks. *Statistics and Computing*, 16:77–92, 2006.

[41] M. Meila and M. I. Jordan. Learning with mixtures of trees. *J. of Machine Learning Research*, 1:1–48, 2000.

[42] Andrew Moore and Weng-Keen Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In *Intl. Conf. on Machine Learning*, pages 552–559, 2003.

[43] Andrew W. Moore and Mary S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *J. of AI Research*, 8:67–91, 1998.

[44] K. Murphy. Active learning of causal Bayes net structure. Technical report, Comp. Sci. Div., UC Berkeley, 2001.

[45] J. Pearl. *Causality: Models, Reasoning and Inference.* Cambridge Univ. Press, 2000.

[46] C. Robert and G. Casella. *Monte Carlo Statisical Methods.* Springer, 2004. 2nd edition.

[47] R. W. Robinson. Counting labeled acyclic digraphs. In F. Harary, editor, *New Directions in the Theory of Graphs*, pages 239–273. Academic Press, 1973.

[48] K. Sachs, O. Perez, D. Pe'er, D. Lauffenburger, and G. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308, 2005.

[49] M. Schmidt, G. Fung, and R. Rosales. Generalized smooth L1 regularization. In *Intl. Conf. on Machine Learning*, 2007. Submitted.

[50] T. Silander and P. Myllmaki. A simple approach for finding the globally optimal Bayesian network structure. In *UAI*, 2006.

[51] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* MIT Press, 2000. 2nd edition.

[52] J. Tian and J. Pearl. Causal discovery from changes. In *UAI*, 2001.

[53] J. Tian and J. Pearl. Causal discovery from changes: a Bayesian approach. Technical report, UCLA, 2001.

[54] S. Tong and D. Koller. Active learning for structure in Bayesian networks. In *Intl. Joint Conf. on AI*, 2001.

[55] I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning*, 2006.

[56] I Tsamardinos, L Brown, and C Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

[57] A. Werhli, M. Grzegorczyk, and D. Husmeier. Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical Gaussian models and Bayesian networks. *Bioinformatics*, 22(20):2523–2531, 2006.

[58] EJ Yeoh, ME Rossa, SA Shurtleff, and WK Williams et al. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1:133–143, 2002.

Part II

Appendices

Appendix A

Software

The code developed and used for this thesis is freely available at `www.cs.ubc.ca/~murphyk/StructureLearning`, and is accompanied by documentation for the major features. The package is named Bayesian Network Structure Learning (BNSL); it was written in Matlab, and consequently can be run on any platform supported by that software. BNSL has no other external dependencies.

BNSL is capable of all the exact and approximate Bayesian structure learning methods reported in this thesis. Both static and dynamic networks can be learned, using multinomial or linear-Gaussian CPDs. All four models of intervention can be used, along with purely observational data. Table A.1 enumerates the major functionality.

Name	Type	Limitation
Dynamic programming [31]	Exact	$d \approxeq 20 - 22$
Exhaustive enumeration	Exact	$d = 6$
Gibbs sampling [37]	Approx.	$d \approxeq 20$
Order space sampling [37]	Approx.	$d \approxeq 20$
Structure space MCMC [13, 24, 35]	Approx.	Depends $(*)$
Optimal DAG [50]	Exact	$d \approxeq 20 - 22$
Local search [27]	Approx.	$d = 500$

Table A.1: Major features of BNSL software

$(*)$ If the global proposal is used, the limitations are the same as dynamic programming; however, if pure local moves are made, the technical limitation becomes $d \approxeq 50$ (in practice, local moves will not mix well for $d \geq 10$).