Journal of Machine Learning Research 11 (2010) 3647-3680 Submitted 6/08; Revised 1/10; Published 12/10

Learning Non-Stationary Dynamic Bayesian Networks

Joshua W. Robinson JOSH@CS.DUKE.EDU
Alexander J. Hartemink AMINK@CS.DUKE.EDU
Department of Computer Science
Duke University
Durham, NC 27708-0129, USA

Editor: Zoubin Ghahramani

Abstract

Learning dynamic Bayesian network structures provides a principled mechanism for identifying conditional dependencies in time-series data. An important assumption of traditional DBN structure learning is that the data are generated by a stationary process, an assumption that is not true in many important settings. In this paper, we introduce a new class of graphical model called a non-stationary dynamic Bayesian network, in which the conditional dependence structure of the underlying data-generation process is permitted to change over time. Non-stationary dynamic Bayesian networks represent a new framework for studying problems in which the structure of a network is evolving over time. Some examples of evolving networks are transcriptional regulatory networks during an organism's development, neural pathways during learning, and traffic patterns during the day. We define the non-stationary DBN model, present an MCMC sampling algorithm for learning the structure of the model from time-series data under different assumptions, and demonstrate the effectiveness of the algorithm on both simulated and biological data.

Keywords: Bayesian networks, graphical models, model selection, structure learning, Monte Carlo methods

1. Introduction

A principled mechanism for identifying conditional dependencies in time-series data is provided through structure learning of dynamic Bayesian networks. An important requirement of this learning is that the time-series data is generated by a distribution that does not change with time—it is stationary. The assumption of stationarity is adequate in many situations since certain aspects of data acquisition or generation can be easily controlled and repeated. However, other interesting and important circumstances exist where that assumption does not hold and potential non-stationarity cannot be ignored.

The inspiration for this model of "networks evolving over time" comes primarily from neurobiology. Dynamic Bayesian networks have been used to identify networks of neural information flow that operate in the brains of songbirds (Smith et al., 2006). When a juvenile male songbird is born, he cannot sing any songs; however, as he ages, he learns songs from other birds singing around him. As the songbird learns from its environment, the networks of neural information flow slowly adapt to make the processing of sensory information more efficient. The analysis carried out by Smith et al. (2006) was limited by the fact that the researchers were forced to learn networks on subsets of the data since DBN structure learning algorithms assume stationarity of the data. Therefore, a learning algorithm based upon dynamic Bayesian networks that relaxes the data stationarity assumption would be ideally suited to this problem.

As another example, structure learning of DBNs has been used widely in reconstructing transcriptional regulatory networks from gene expression data (Friedman et al., 2000; Hartemink et al., 2001). But during development, these regulatory networks are evolving over time, with certain conditional dependencies between gene products being created as the organism develops, and others being destroyed. As yet another example, one can use a DBN to model traffic flow patterns. The roads upon which traffic passes do not change on a daily basis, but the dynamic utilization of those roads changes daily during morning rush, lunch, evening rush, and weekends.

If one collects time-series data describing the levels of gene products in the case of transcriptional regulation, traffic density in the case of traffic flow, or neural activity in the case of neural information flow, and attempts to learn a DBN describing the conditional dependencies in the time-series, one could be seriously misled if the data-generation process is non-stationary.

Here, we introduce a new class of graphical model called a non-stationary dynamic Bayesian network (nsDBN), in which the conditional dependence structure of the underlying data-generation process is permitted to change over time. In the remainder of the paper, we introduce and define the nsDBN framework, present a simple but elegant algorithm for efficiently learning the structure of an nsDBN from time-series data under a variety of different assumptions, and demonstrate the effectiveness of these algorithms on both simulated and experimental data.

1.1 Related Work

Representing relationships or statistical dependencies between variables in the form of a network is a popular technique in many communities, from economics to computational biology to sociology. Recently, researchers have been interested in elucidating the temporal evolution of genetic regulatory networks (Arbeitman et al., 2002; Luscombe et al., 2004) and neural information flow networks (Smith et al., 2006), but were forced to perform their analysis on subsets of the data since their structure learning algorithms assumed stationarity of the data. Additionally, some economics techniques require prior specification of a graphical model (for a particular data set) and assume it is stationary (Carvalho and West, 2007) when the data set may actually be highly non-stationary (Xuan and Murphy, 2007). Therefore, the identification of non-stationary behavior in graphical models is of significant interest and importance to many communities.

We divide models of evolving statistical dependencies into those that model changing structures and those that model changing parameters, and describe examples of both in this section. Additionally, we briefly explain how our approach compares and some of the advantages it provides over other methods.

1.1.1 STRUCTURAL NON-STATIONARITY

Approaches that learn structural non-stationarities are those that explicitly model the presence of statistical dependencies between variables and allow them to appear and disappear over time (e.g., they define directed or undirected networks whose edges change over time).

In recent work modeling the temporal progression of networks from the social networks community (Hanneke and Xing, 2006), the $p*$ or exponential random graph model (ERGM) (Wasserman and Pattison, 1996) was generalized to the temporal ERGM (tERGM) model, where a structural evolutionary process is modeled with a set of features between adjacent network structures (Hanneke and Xing, 2006). The tERGM model was further extended to the hidden tERGM (htERGM) to handle situations where the networks are latent (unobserved) variables that generate observed time-series data (Guo et al., 2006, 2007).

While the transition model of an htERGM allows for nearly unlimited generality in characterizing how the network structure changes with time, it is restricted to functions of temporally adjacent network structures. Therefore, an evolutionary process that differs between early observations and later observations may not be effectively captured by a single transition model. Also, the emission model defined in Guo et al. (2007) must be estimated *a priori* and only captures pairwise correlations between variables; more complicated relationships between multiple variables that change over time may be missed altogether. Finally, Guo and colleagues focus on identifying undirected edges. Although it is possible to adapt the ERGM model for directed graphs, it becomes more difficult to define the parameters of the tERGM and the emission model assuming directed edges.

In the continuous domain, some research has focused on learning the structure of a time-varying undirected Gaussian graphical model (Talih and Hengartner, 2005). These authors use a reversible-jump MCMC approach to estimate the time-varying variance structure of the data. They explicitly model the network's edges as non-zeroes in the precision matrix. While this model allows for fast, efficient sampling, it only does so by defining several restrictions to the model space. First, the structural evolutionary process is piecewise-stationary and restricted to single edge changes. Rapid and significant structural changes would be approximated by a slowly changing network structure, resulting in an inaccurate portrayal of the true evolutionary behavior of the data-generating process. Additionally, the total number of segments or epochs in the piecewise stationary process is assumed known *a priori*, thereby limiting application of the method in situations where the number of epochs is not known. Finally, this approach only identifies undirected edges (correlations), while time-series data should allow one to identify directed edges (conditional dependencies).

A similar algorithm—also based on undirected Gaussian graphical models—has been developed to segment multivariate time-series data (Xuan and Murphy, 2007). This approach iterates between a convex optimization for determining the graph structure and a dynamic programming algorithm for calculating the segmentation. This results in some notable advantages over Talih and Hengartner (2005): it has no single edge change restriction and the number of segments is calculated *a posteriori*. The main restriction, however, is that the graph structure must be decomposable. Additionally, because this method models structure as non-zeroes in the precision matrix, it only identifies undirected edges. Finally, the networks (precision matrices) in each segment are assumed independent, preventing the sharing of parameters and data between segments.

Another recently proposed model for identifying evolving networks is the temporally smoothed l_1-regularized logistic regression (TESLA) method, described by Kolar et al. (2010) and Ahmed and Xing (2009). This approach involves learning a discrete binary Markov random field with sparse, varying parameters. A useful advantage of this model is that can be reparameterized to account for either smoothly varying or abrupt sudden changes to the network structure. Additionally, it scales well to thousands of nodes. However, it is undirected, requires binary data, and only captures pairwise relationships between variables. The binary input and pairwise relationship requirements may be relaxed, but the likelihood function would have to be significantly modified to incorporate cliques; the resulting optimization problem might become intractable.

1.1.2 PARAMETER NON-STATIONARITY

Approaches that learn parameter non-stationarities are those that explicitly model the evolution of parameters over time. Here we only focus on a few that can be represented as networks with temporally changing parameters.

Switching state-space models (SSMs) represent a piecewise-stationary extension of linear dynamic systems (Ghahramani and Hinton, 2000). In an SSM, a sequence of observations is modeled as a function of several independent hidden variables which is itself controlled by a switch variable. The hidden variables as well as the switch variable all have Markovian dynamics. While this approach is similar to ours in that it describes the evolution of a piecewise-stationary process, it does have some notable differences. Our model has no hidden variables, only observed variables. Critically, our variables are not assumed to be independent; rather, their dependencies are unknown and must be estimated *a posteriori*. Additionally, the piecewise-stationary process in our approach does not follow Markovian dynamics, like the switch variable in an SSM.

A more closely related model is the recently published non-homogeneous Bayesian network (Grzegorczyk et al., 2008). In this model, the conditional distributions of the variables are assumed to follow a mixture of Gaussians. Each observation is allocated to a single mixture component where the parameters and number of the mixture components are determined *a posteriori* using an allocation sampler. Unfortunately, an allocation sampler does not allow data to be shared across different mixture components since they are assumed independent. However, this model seems to do a good job of capturing the non-stationary behavior of parameters in a Gaussian Bayesian network, assuming that the underlying structure is inferred correctly.

1.1.3 OUR APPROACH

Although we provide more detail in Section 3, for the purpose of comparison, we define our approach as the identification of a discrete Bayesian network that evolves according to a piecewise-stationary process where edges are gained and lost over time. Building on the Bayesian network model allows us to identify directed networks and results in efficient learning (under certain assumptions). Additionally, when the conditional probability distributions of the Bayesian network are multinomial, we can identify linear, non-linear, and combinatorial interactions between variables. Finally, the piecewise-stationary assumption (and additional constraints on how and when edge changes occur) allows our method to scale to large data sets with many variables and provides a natural parameterization for placing priors on the structural evolution process.

Our method falls into the category of models that identify non-stationarities in structure, not parameters. In the rest of this paper, we define non-stationarities as times at which conditional dependencies between variables are gained or lost (i.e., edges are gained or lost). We have chosen to focus on structural non-stationarity for several reasons. First, we are not as interested in making predictions about future data (such as might be the case with spam prediction) as we are in the analysis of collected data to identify non-stationary statistical relationships between variables in multivariate time-series. Additionally, the problems we analyze in this paper are highly multimodal in the posterior over structures and likely to be even more varied in the posterior over parameters. By assuming conjugate parameter priors, we address both problems by averaging out all possible parameters and only examining the posterior over structures.

However, we recognize that the parameters of the conditional distributions may also change with time. Some changes to the parameters of the conditional distributions may effectively result in a structural change, while other changes may be dramatic, yet not alter the structure. Ultimately, a trade-off must be made between simplifying model assumptions resulting in greater statistical power versus a completely general framework requiring approximation schemes. Under our modeling assumptions, we can identify non-stationarities in the parameters of the conditional distributions that are significant enough to result in structural changes; we assume other changes are small enough to not alter edges in the predicted structure.

2. Brief Review of Structure Learning of Bayesian Networks

Bayesian networks are directed acyclic graphical models that represent conditional dependencies between variables as edges. They define a simple decomposition of the complete joint distribution—a variable x_i is conditionally independent of its non-descendants given its parents. Therefore, the joint distribution of every variable x_i can be rewritten as $P(x_1,\dots,x_n) = \prod_i P(x_i|\pi_i)$, where π_i are the parents of x_i. Bayesian networks may be learned on time-series data, but the semantics are slightly different, leading to the dynamic Bayesian network (DBN) model. DBNs enable cyclic dependencies between variables to be modeled across time. DBNs are a special case of Bayesian networks, with modeling assumptions regarding how far back in time one variable can depend on another (minimum and maximum lag), and constraints placed on edges so that they do not go backwards in time. For notational simplicity, we assume hereafter that the minimum and maximum lag are both 1.

The task of inferring the structure (i.e., the set of conditional dependencies) of a Bayesian network is typically expressed using Bayes' rule, where the posterior probability of a given network structure G (i.e., the set of conditional dependencies) after having observed data D is given by

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)}.$$

Since $P(D)$ is the same for all structures, we see that $P(G|D) \propto P(D|G)P(G)$. The prior over networks $P(G)$ can be used to incorporate prior knowledge about the existence of specific edges (Bernard and Hartemink, 2005) or the overall topology of the network (e.g., sparse); often, prior knowledge is not available and $P(\Theta)$ is assumed uniform. The marginal likelihood $P(D|G)$ is further defined in terms of the parameters Θ_i of the conditional probability distributions (CPDs) between a variable and its parents $P(x_i|\pi_i,\Theta_i)$. The entire set of parameters Θ_i for all variables is simply Θ.

$$P(D|G) = \int P(D|\Theta,G)P(\Theta|G)\mathrm{d}\Theta. \tag{1}$$

One might be interested in inferring the values of Θ given a particular network, but we will be focusing on learning the network itself, or the set of conditional dependencies. In computer science, this task is often referred to as structure learning; in statistics, it is often called model selection. More detailed reviews of structure learning of Bayesian networks can be found in Buntine (1996), Chrisman (1998), Krause (1998), and Murphy (2001).

For the remainder of this section, we review the most common approaches to inferring dynamic Bayesian networks, thereby setting the stage for what changes we will need to make in the following section to handle inference in the non-stationary setting. For brevity, we will focus on dynamic Bayesian networks with discrete variables, though the ideas could also be applied to networks of continuous variables. Details specific to structure learning of Bayesian networks with continuous variables can be found in Hofmann and Tresp (1995) and Margaritis (2005).

2.1 Evaluating the Marginal Likelihood $P(D|G)$

Evaluation of the marginal likelihood in Equation (1) can be performed either approximately or exactly. The marginal likelihood can be approximated with the Akaike information criterion (AIC), the Bayesian information criterion (BIC) (Friedman et al., 1998), or the minimum description length (MDL) (Lam and Bacchus, 1994; Suzuki, 1996) metric. Each of these metrics suggests how model complexity should be penalized. For example, the AIC metric penalizes free parameters less strongly than the BIC metric; therefore, a Bayesian network learned using the AIC metric would be likely be more dense than a Bayesian network learned using the BIC metric.

Alternatively, if a conjugate prior is placed on Θ which is then marginalized out, the value of $P(D|G)$ can be computed exactly. Assuming that Θ parameterizes multinomially distributed conditional probability distributions, one can place a conjugate Dirichlet prior on the parameters Θ and then marginalize them out to obtain the Bayesian-Dirichlet (BD) metric. The BD metric can be further modified so that all of the networks that represent the same set of conditional independence relationships have the same probability; this is called the Bayesian-Dirichlet equivalent (BDe) metric (Heckerman et al., 1995). By using a Dirichlet prior and marginalizing over all values of Θ, the BD and BDe metrics inherently penalize more complex models, so a prior on the network $P(G)$ may not always be necessary. The primary advantage of the BD family of metrics is that the evaluation of $P(D|G)$ is exact and can be computed in closed form. However, if the assumption that the parameters of CPDs are multinomially distributed is incorrect, these metrics might not find the true network of conditional dependencies.

Since we will be modifying it later in this paper, we show the closed-form expression for the BDe metric below:

$$P(D|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})} \tag{2}$$

where q_i is the number of configurations of the parent set π_i, r_i is the number of discrete states of variable x_i, N_{ijk} is the number of times x_i took on the value k given the parent configuration j, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, and α_{ij} and α_{ijk} are Dirichlet hyper-parameters on various entries in Θ. If α_{ijk} is set to $\alpha/(q_i r_i)$ (essentially a uniform Dirichlet prior), we get a special case of the BDe metric: the uniform BDe metric (BDeu), whose parameter priors are all controlled by a single hyperparameter α.

2.2 Deciding Between Search or Sampling Strategies

Once a form of the marginal likelihood $P(D|G)$ is defined and a method for evaluating it is chosen, one must decide whether the objective is to identify the best network or to capture the uncertainty over the space of all posterior networks. Search methods may be used to find the best network or set of networks (i.e., a mode in the space of all networks), while sampling methods may be used to estimate posterior quantities from the space of all networks.

Search methods may be exact or heuristic, but exact search for Bayesian networks is computationally infeasible for more than about 30 variables because the number of possible networks is super-exponential in the number of variables. In fact, identifying the highest scoring Bayesian network has been shown to be NP-Hard (Chickering et al., 1994). If the maximum number of parents for any variable is limited to some constant p_{max}, the total number of valid networks becomes exponential in the number of variables; however, finding the best network is still NP-Hard (for $p_{max} \geq 2$). Therefore, heuristic searches are often employed, including greedy hill-climbing, simulated annealing (Heckerman et al., 1995), the K2 algorithm (Chickering et al., 1995), genetic algorithms (Larrañaga et al., 1996), and even ant colony optimization (de Campos et al., 2002). Most heuristic searches perform well in a variety of settings, with greedy hill-climbing and simulated annealing being the most commonly used techniques.

A different approach from search is the use of sampling techniques (Madigan et al., 1995; Giudici et al., 1999; Tarantola, 2004). If the best network is all that is desired, heuristic searches will typically find it more quickly than sampling techniques. However, sampling methods allow the probability or importance of individual edges to be evaluated over all possible networks. In settings where many modes are expected, sampling techniques will more accurately capture posterior probabilities regarding various properties of the network. The primary disadvantage of sampling methods in comparison to search methods is that they often take longer before accurate results become available.

A common sampling technique often used in this setting is the Metropolis-Hastings algorithm, which is a Markov chain Monte Carlo (MCMC) method. The M-H acceptance probability for moving from state x to state x' is shown below, where each *state* is a DBN.

$$\alpha(x,x') = \min\left\{1, \frac{p(D|x')}{p(D|x)} \times \frac{p(x' \to x)}{p(x \to x')}\right\} = \min\left\{1,\ \underbrace{\frac{p(D|x')}{p(D|x)}}_{\textit{likelihood ratio}} \times \underbrace{\frac{\sum_{M'} p(M')p(x|x',M')}{\sum_M p(M)p(x'|x,M)}}_{\textit{proposal ratio}}\right\} \tag{3}$$

where M is the move type that allows for a transition from state x to x' and M' is the reverse move type for a transition from state x' back to state x. While multiple moves may result in a transition from state x to state x' (and vice versa), typically there is only a single move for each transition. In such a case, the sums over M and M' each only include one term, and the proposal ratio can be split into two terms: one is the ratio of the proposal probabilities for move types and the other is the ratio of selecting a particular state given the current state and the move type. The choice of scoring metric determines the likelihoods, and $p(M')$ and $p(M)$ are often chosen *a priori* to be equivalent or simple to calculate.

2.3 Determining the Move Set

Once a search or sampling strategy has been selected, one must determine how to move through the space of all networks. A *move set* defines a set of local traversal operators for moving from a particular state (i.e., a network) to nearby states. The set of states that can be reached from the current one is often called the *local neighborhood* of that state. The values of $p(x'|x,M)$ and $p(x|x',M)$ in the proposal ratio are defined by the move set.

Ideally, the move set includes changes that allow posterior modes to be frequently visited. For example, it is reasonable to assume that networks that differ by a single edge will have similar likelihoods. A well-designed move set results in fast convergence since less time is spent in the low probability regions of the state space. For example, with traditional Bayesian networks, Madigan et al. (1995) proposed that the move set be: *add an edge* and *delete an edge*. However, it was later shown by Giudici and Castelo (2003) that the convergence rate could be increased with the addition of another move to the move set: *reverse an edge*.

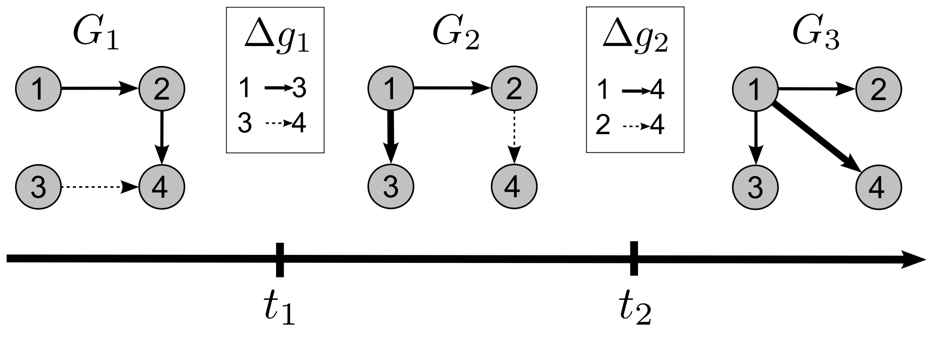


Figure 1: An example nsDBN with labeled components (namely, transition times t_1 and t_2 and edge change sets Δg_1 and Δg_2). Edges that are gained from the previous epoch are shown as thicker lines and edges that will be lost in the next epoch are shown as dashed lines. Note that the network at each epoch is actually a DBN drawn in compact form, where each edge represents a statistical dependence between a node at time t and its parent at the previous time $t-1$.

3. Learning Non-Stationary Dynamic Bayesian Networks

While DBNs are excellent models for describing dependencies between time-series random variables, they cannot represent or reason about how these dependencies might change over time. In contrast, our nsDBN model is capable of characterizing dependencies between temporally observed variables, as well as reasoning about whether and how those dependencies change. Because DBNs are well studied and well understood, we have chosen to introduce the details of our nsDBN model by building upon the existing DBN model. Therefore, we use this section to detail how the structure learning procedure for DBNs needs to be modified and extended to account for non-stationarity when learning a non-stationary DBN.

Assume that we observe the state of n random variables at N discrete times. Call this multivariate time-series data D, and further assume that it is generated according to a non-stationary process, which is unknown. The process is non-stationary in the sense that the network of conditional dependencies prevailing at any given time is itself changing over time. We call the initial (dynamic) network of conditional dependencies G_1 and subsequent networks are called $G_2, G_3, \ldots, G_m$. We define Δg_i to be the set of edges that change (either added or deleted) between G_i and G_{i+1}. The number of edge changes specified in Δg_i is s_i. We define the *transition time* t_i to be the time at which G_i is replaced by G_{i+1} in the data-generation process. We call the period of time between consecutive transition times—during which a single network of conditional dependencies is operative—an *epoch*. So we say that G_1 prevails during the first epoch, G_2 prevails during the second epoch, and so forth. We will refer to the entire series of prevailing networks as the *structure* of the nsDBN. Figure 1 shows an example nsDBN with the components labeled.

3.1 Updating the Marginal Likelihood and Incorporating Priors

Since we wish to estimate a set of networks instead of one network we must derive a new expression for the marginal likelihood. Consider the simplest case where $m = 2$ and the transition time t_1 at which the structure of the nsDBN evolves from network G_1 to network G_2 is known. We would like to find the networks G_1 and G_2 that have the highest probability given the observed time-series data D and prior. Thus, we want to find the networks that maximize the expression below:

$$P(G_1, G_2|D, t_1) = \frac{P(D|G_1, G_2, t_1)P(G_1, G_2|t_1)}{P(D|t_1)} \propto P(D|G_1, G_2, t_1)P(G_1, G_2|t_1).$$

To maximize the marginal likelihood $P(D|G_1, G_2, t_1)$ in the above expression, one approach might be to estimate a different network for each epoch. Unfortunately, if the number of observations in each epoch is small, accurate reconstruction of the correct structure may be difficult or impossible (Friedman and Yakhini, 1996; Smith et al., 2003). Additionally, learning each network separately might lead to predictions that are vastly different during each epoch. If prior knowledge about the problem dictates that the networks will not vary dramatically across adjacent epochs, information about the networks learned in adjacent epochs can be leveraged to increase the accuracy of the network learned in the current epoch.

Expanding the simple formulation to multiple epochs, assume there exist m different epochs with transition times $T = \{t_1, \ldots, t_{m-1}\}$. The network G_{i+1} prevailing in epoch $i+1$ differs from network G_i prevailing in epoch i by a set of edge changes we call Δg_i. We would like to determine the sequence of networks $G_1, \ldots, G_m$ that maximize the posterior given below:

$$P(G_1, \ldots, G_m|D, T) \propto P(D|G_1, \ldots, G_m, T)P(G_1, \ldots, G_m|T). \tag{4}$$

Since each network differs from the previous one by a set of edge changes, we can rewrite the prior and obtain the expression below:

$$P(G_1, \ldots, G_m|T) = P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}|T).$$

By writing the objective function this way, we rephrase the problem as finding the initial network and $m-1$ sets of edge changes that best describe the data. Unfortunately, the search space for finding the best structure is still super-exponential (in n) for the initial network and the task of identifying the other networks is exponential (in both m and n). However, using this formulation, it is easy to place some constraints on the search space to reduce the number of valid structures. If we specify that the maximum number of parents for any variable is p_{max} and, through priors, require that the maximum number of edge changes for any Δg_i is s_{max}, the search space for finding the best structure becomes exponential (in n) for the initial network and exponential (in m and s_{max}) for the other networks.

We assume the prior over networks can be further split into components describing the initial network and subsequent edge changes, shown below:

$$P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}|T) = P(G_1)P(\Delta g_1, \ldots, \Delta g_{m-1}|T).$$

leading to the final form of the posterior

$$P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}|D, T) \propto P(D|G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T)P(G_1)P(\Delta g_1, \ldots, \Delta g_{m-1}|T). \tag{5}$$

As in the stationary setting, if prior knowledge about particular edges or overall topology is available, an informative prior can be placed on $P(G_1)$. In the context of the simulations and experiments in this paper, we had no prior knowledge about the network; thus, we assumed a uniform prior on $P(G_1)$. We do, however, place some prior assumptions on the ways in which edges change in the structure. First, we assume that the networks evolve smoothly over time. To encode this prior knowledge, we place a truncated geometric prior with parameter $p = 1 - \mathrm{e}^{-\lambda_s}$ on the number of changes (s_i) in each edge change set (Δg_i), with the distribution truncated for $s_i > s_{max}$. For the problems we explore, the truncation has no noticeable effect since s_{max} is chosen to be large. The updated posterior for the structure is given below:

$$\begin{aligned} P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}|D, T) &\propto P(D|G_1, \ldots, G_m, T) \prod_i \frac{\left(1 - \mathrm{e}^{-\lambda_s}\right)\left(\mathrm{e}^{-\lambda_s}\right)^{s_i}}{1 - \left(\mathrm{e}^{-\lambda_s}\right)^{s_{max}+1}} \\ &\propto P(D|G_1, \ldots, G_m, T) \prod_i \left(\mathrm{e}^{-\lambda_s}\right)^{s_i} \\ &\propto P(D|G_1, \ldots, G_m, T)\mathrm{e}^{-\lambda_s s}, \end{aligned}$$

where $s = \sum_i s_i$. Therefore, a (truncated) geometric prior on each s_i is essentially equivalent to a (truncated) geometric prior on the sufficient statistic s, the total number of edge changes.

When the transition times are *a priori* unknown, we would like to estimate them. The posterior in this setting becomes

$$\begin{aligned} P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T|D) &\propto P(D|G_1, \ldots, G_m, T)P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T) \\ &= P(D|G_1, \ldots, G_m, T)P(G_1)P(\Delta g_1, \ldots, \Delta g_{m-1}, T) \\ &= P(D|G_1, \ldots, G_m, T)P(G_1)P(\Delta g_1, \ldots, \Delta g_{m-1})P(T). \end{aligned}$$

We assume that the joint prior over the evolutionary behavior of the network and the locations of transition times can be decomposed into two independent components. We continue to use the previous geometric prior with parameter $p = 1 - \mathrm{e}^{-\lambda_s}$ on the total number of edge changes. Any choice for $P(T)$ can be made, but for the purposes of this paper, we have no prior knowledge about the transition times; therefore, we assume a uniform prior on T so that all settings of transition times are equally likely for a given value of m.

Finally, when neither the transition times nor the number of epochs are known *a priori*, both the number and times of transitions must be estimated *a posteriori*. If prior knowledge dictates that the networks evolve slowly over time (i.e., a transition does not occur at every observation), we can include this knowledge by placing a non-uniform prior on m, for example a (truncated) geometric prior with parameter $p = 1 - \mathrm{e}^{\lambda_m}$ on the number of epochs m, with the distribution truncated for $m > N$. A geometric prior on the number of epochs is equivalent to a geometric prior on epoch lengths (see Appendix A).

The form of the geometric prior was chosen for convenience since under a log transformation, the likelihood (and prior) calculations reduce to: $\log(\text{likelihood}) - \lambda_s s - \lambda_m m$. In general, large values of λ_m encode the strong prior belief that the structure changes slowly (i.e., few epochs exist) and large values of λ_s encode the strong prior belief that the structure changes smoothly (i.e., few edge changes exist).

Fortunately, the likelihood component of the posterior does not change whether we know the transition times *a priori* or not. Therefore, any uncertainty about the transition times can be incorporated into the evaluation of the prior, leaving the evaluation of the likelihood unchanged.

3.2 Evaluating the New Marginal Likelihood

Now that a new likelihood has been defined, we must decide on which method to use for evaluation and how to modify it to account for multiple structures. Any of the previously mentioned metrics can be modified to account for non-stationarity, but we chose to extend the BDe metric because it is exact and because edges are the only representation of conditional dependencies that are left after the parameters have been marginalized away. This provides a useful paradigm for non-stationarity that is both simple to define and easy to analyze.

The BDe metric needs to be modified when learning an nsDBN because in Equation (2), N_{ij} and N_{ijk} are calculated for a particular parent set over the entire data set D. However, in an nsDBN, a node may have multiple parent sets operative at different times. The calculation for N_{ij} and N_{ijk} must therefore be modified to specify the *intervals* during which each parent set is operative. Note that an interval may include several epochs. An epoch is defined between adjacent transition times while an interval is defined as the union of consecutive epochs during which a particular parent set is operative (which may include all epochs).

For each node i, the parent set π_i in the BDe metric is replaced by a set of parent sets π_{ih}, where h indexes the interval I_h during which parent set π_{ih} is operative for node i. Letting p_i be the number of such intervals and q_{ih} be the number of configurations of π_{ih} results in the expression below:

$$P(D|G_1,\ldots,G_m,T) \propto \prod_{i=1}^{n}\prod_{h=1}^{p_i}\prod_{j=1}^{q_{ih}} \frac{\Gamma(\alpha_{ij}(I_h))}{\Gamma(\alpha_{ij}(I_h)+N_{ij}(I_h))} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}(I_h)+N_{ijk}(I_h))}{\Gamma(\alpha_{ijk}(I_h))}$$

where the counts N_{ijk} and pseudocounts α_{ijk} have been modified to apply only to the data in each interval I_h. The modified BDe metric will be referred to as nsBDe. Given that $|I_h| = t_{h+1} - t_h$, we set $\alpha_{ijk}(I_h) = (\alpha_{ijk}|I_h|)/N$ (e.g., proportional to the length of the interval during which that particular parent set is operative). If α_{ijk} is set everywhere to $\alpha/(q_i r_i)$ as in the BDeu metric, then we generalize the BDeu metric to nsBDeu, and the parameter priors are again all controlled by a single hyperparameter α.

The derivation of the BDe metric requires seven assumptions: multinomial sample, parameter independence, likelihood modularity, parameter modularity, Dirichlet distribution for parameters, complete data, likelihood equivalence, and structure possibility (Heckerman et al., 1995). To extend the BDe metric to account for non-stationary behavior, we must extend only the parameter independence assumption.

Parameter independence is split into local parameter independence and global parameter independence. Letting G represent an individual network, global parameter independence is represented as $p(\Theta_G|G) = \prod_{i=1}^{n} p(\Theta_i|G)$. In other words, the conditional probabilities can be decomposed by variable. Local parameter independence is represented as $p(\Theta_i|G) = \prod_{j=1}^{q_i} p(\Theta_{ij}|G)$; the conditional probabilities for each variable are decomposable by parent configuration.

For nsDBNs, we also need to assume parameter independence across intervals. Parameter independence is thus split into three assumptions. The updated parameter independence assumptions are defined below, where $\mathbb{G}$ represents an nsDBN structure, or set of networks.

Global parameter independence: The conditional probabilities are decomposable by variable:

$$p(\Theta_{\mathbb{G}}|\mathbb{G}) = \prod_{i=1}^{n} p(\Theta_i|\mathbb{G}).$$

Interval parameter independence: The conditional probabilities for each variable are decomposable by interval:

$$p(\Theta_i|\mathbb{G}) = \prod_{h=1}^{p_i} p(\Theta_{ih}|\mathbb{G}).$$

Local parameter independence: The conditional probabilities for each variable for each interval are decomposable by parent configuration:

$$p(\Theta_{ih}|\mathbb{G}) = \prod_{j=1}^{q_{ih}} p(\Theta_{ihj}|\mathbb{G}).$$

3.3 Using a Sampling Strategy and Choosing a New Move Set

We decide to take a sampling approach rather than using heuristic search because the posterior over structures includes many modes. In particular, when the transition times are not known *a priori*, the posterior is highly multimodal because structures with slightly different transition times likely have similar posterior probabilities. Additionally, sampling offers the further benefit of allowing us to evaluate interesting posterior quantities, such as when are the most likely times at which transitions occur—a question that would be difficult to answer in the context of heuristic search.

Because the number of possible nsDBN structures is so large (significantly greater than the number of possible DBNs), we must be careful about what is included in the move set. To provide quick convergence, we want to ensure that every move in the move set efficiently jumps between posterior modes. Therefore, the majority of the next section is devoted to describing effective move sets under different levels of uncertainty.

4. Settings

Each of the following subsections demonstrates a method for calculating an nsDBN under a variety of *settings* that differ in terms of the level of uncertainty about the number and times of transitions. The different settings will be abbreviated according to the type of uncertainty: whether the *number* of transitions is *known* (KN) or *unknown* (UN) and whether the transition *times* themselves are *known* (KT) or *unknown* (UT). Figure 2 shows plate diagrams relating the DBN model to the three different settings of the nsDBN model, described in the following subsections.

4.1 Known Number and Known Times of Transitions (KNKT)

In the KNKT setting, we know all of the transition times *a priori*; therefore, we only need to identify the most likely initial network G_1 and sets of edge changes $\Delta g_1, \ldots, \Delta g_{m-1}$. Thus, we wish to maximize Equation (5).

To create a move set that results in an effectively mixing chain, we consider which types of local moves result in jumps between posterior modes. As mentioned earlier, networks that differ by a single edge will probably have similar likelihoods. Therefore, the move set includes a single edge addition or deletion to G_1. Each of these moves results in the structural difference of a single edge over all observations. One can also consider adding or deleting an edge in a particular Δg_i; this results in the structural difference of a single edge for all observations after t_i. Finally, we consider moving an edge from one Δg_i to another, which results in the structural difference of a single edge for all observations between t_i and t_j. These moves are listed as M_1–M_5 in Table 1, along with the various proposal probabilities defined in the Metropolis-Hastings acceptance ratio shown in Equation (3).

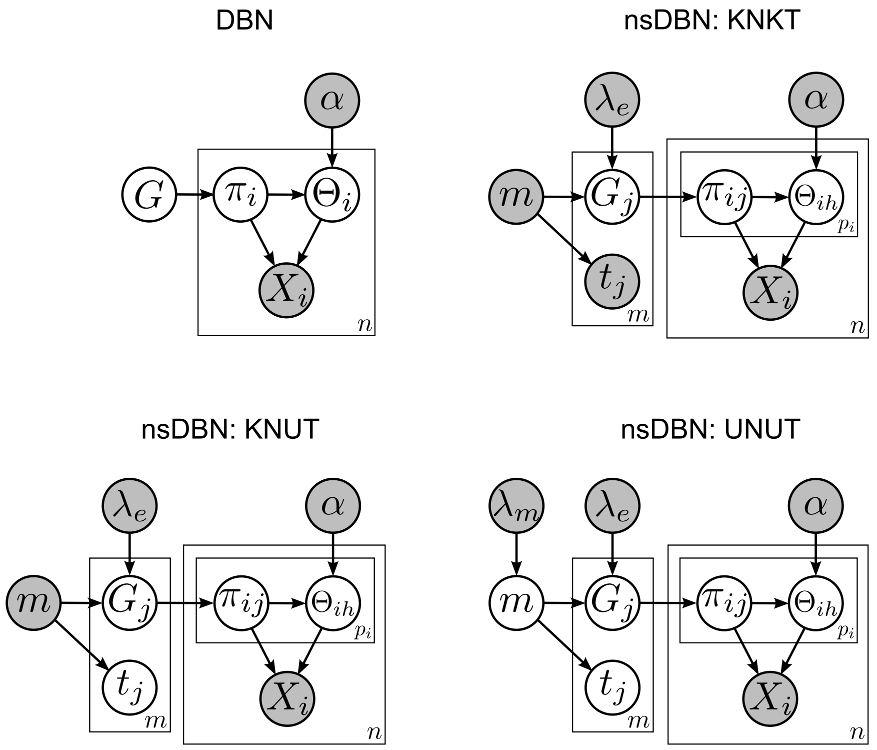


Figure 2: Plate diagrams relating DBNs to nsDBNs under each setting. Gray circles represent quantities that are known *a priori* while white circles represent those that are unknown. Non-stationary DBNs may have multiple networks (m, indexed by j) and multiple different parent sets for each variable (p_i for variable i, indexed by h). In the KNKT setting, the transition times t_j are known *a priori*, but they must be estimated in the two other settings. In the KNUT setting, the number of epochs m is still known, but the transition times themselves are not. Finally, in the UNUT setting, even the number of epochs is unknown; instead a truncated geometric prior is placed on m.

4.2 Known Number But Unknown Times of Transitions (KNUT)

Knowing in advance the times at which all the transitions occur, as was assumed in the previous subsection, is often unrealistic. To relax this assumption, we now assume that while m is known, the set T is not given *a priori* but must also be estimated. Thus, rather than maximizing Equation (5), we maximize the expression below:

$$P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T|D).$$

Structures with the same edge sets but slightly different transition times will probably have similar likelihoods. Therefore, we can add a new move that proposes a local shift to one of the transition times: let d be some small positive integer and let the new time t'_i be drawn from a discrete uniform distribution $t'_i \sim DU(t_i - d, t_i + d)$ with the constraint that $t_{i-1} < t'_i < t_{i+1}$. Initially, we set the $m-1$ transition times so that the epochs are roughly equal in length. This placement allows the transition times ample room to locally shift without "bumping" into each other too early in the sampling procedure. The complete move set for this setting includes all of the moves described previously as well as this new *local shift move*, listed as M_6 in Table 1.

As with the last setting, the number of epochs does not change; therefore, only the prior on the number of edge changes s is used.

4.3 Unknown Number and Unknown Times of Transitions (UNUT)

In the most general UNUT setting, both the transition times T and the number of transitions are unknown and must be estimated. While this is the most interesting setting, it is also the most difficult. Since the move set from the KNUT setting provides a solution to this problem when m is known, a simple approach would be to try various values of m and then determine which value of m seems optimal. However, this approach is theoretically unsatisfying and would be incredibly slow. Instead, we will further augment the move set to allow the number of transitions to change. Since both the number of edge changes s and the number of epochs m are allowed to vary, we need to incorporate both priors mentioned in Section 3.1 when evaluating the posterior.

To allow the number of epochs m to change during sampling, we introduce *merge* and *split* operations to the move set. For the merge operation, two adjacent edge sets (Δg_i and Δg_{i+1}) are combined to create a new edge set. The transition time of the new edge set is selected to be the mean of the previous locations weighted by the size of each edge set: $t'_i = (s_i t_i + s_{i+1} t_{i+1})/(s_i + s_{i+1})$. For the split operation, an edge set Δg_i is randomly chosen and randomly partitioned into two new edge sets $\Delta g'_i$ and $\Delta g'_{i+1}$ with all subsequent edge sets re-indexed appropriately. Each new transition time is selected as described above. The move set is completed with the inclusion of the *add transition time* and *delete transition time* operations. These moves are similar to the split and merge operations except they also increase or decrease s, the total number of edge changes in the structure. The four additional moves are listed as M_7–M_{10} in Table 1.

4.4 MCMC Sampler Implementation Details

In practice, the sampler is designed so that the proposal ratio $\frac{p(M')}{p(M)}\frac{p(x|x',M')}{p(x'|x,M)}$ is exactly 1 for most moves. For example, if either move M_1 or move M_2 is randomly selected, the sampling procedure is as follows: random variables x_i and x_j are selected, if the edge $x_i \to x_j$ exists in G_1, it is deleted, otherwise it is added (subject to the maximum of p_{max} parents constraint). We know that the maximal number of edges in G_1 is np_{max} (due to the maximum parent constraint) and we let E_1 be the current number of edges in G_1. If we are making move M_1, the probability that we select a legal edge to add is $P_a = \frac{np_{max}-E_1}{np_{max}}$. The probability of making the reverse move (from x' back to x) is $P_d = \frac{E_1+1}{np_{max}}$. The resulting proposal ratio is thus:

$$\frac{P_d}{P_a}\frac{p(x|x',M'_2)}{p(x'|x,M_1)} = \frac{E_1+1}{np_{max}-E_1}\frac{np_{max}-E_1}{E_1+1} = 1.$$

A similar approach can be applied to the other moves.

This paradigm also handles boundary cases: if G_1 is complete, then $P(M_1)=0$; if G_1 is empty, then $P(M_2)=0$; if $s_i = s_{max}\forall i$, then $P(M_3)=0$; if $s_i = 1\forall i$, then $P(M_4)=0$; etc.

The relative proposal probabilities between different moves are designed so that all pairs of complementary moves (a move and its reverse move) are equally likely. Under the KNKT setting, $P_a+P_d = P_{ae}+P_{de} = 2P_{me}$. Under the KNUT setting, $P_a+P_d = P_{ae}+P_{de} = 2P_{me} = 2P_{st}$. Finally, under the UNUT setting, $P_a+P_d = P_{ae}+P_{de} = 2P_{me} = 2P_{st} = P_m+P_s = P_{ag}+P_{dg}$.

Move type M	Proposal probability	$\frac{p(M')}{p(M)}$	$\frac{p(x\|x',M')}{p(x'\|x,M)}$			
(M_1) add edge to G_1	P_a	$\frac{P_d}{P_a}$	$\frac{(E_1+1)^{-1}}{(np_{max}-E_1)^{-1}} = \frac{np_{max}-E_1}{E_1+1}$			
(M_2) delete edge from G_1	P_d	$\frac{P_a}{P_d}$	$\frac{(np_{max}-E_1+1)^{-1}}{E_1^{-1}} = \frac{E_1}{np_{max}-E_1+1}$			
(M_3) add edge to Δg_i	P_{ae}	$\frac{P_{de}}{P_{ae}}$	$\frac{m^{-1}(S_i+1)^{-1}}{m^{-1}(S_{max}-S_i)^{-1}} = \frac{S_{max}-S_i}{S_i+1}$	KNKT	KNUT	
(M_4) delete edge from Δg_i	P_{de}	$\frac{P_{ae}}{P_{de}}$	$\frac{m^{-1}(S_{max}-S_i+1)^{-1}}{m^{-1}S_i^{-1}} = \frac{S_i}{S_{max}-S_i+1}$			
(M_5) move edge from Δg_i to Δg_j	P_{me}	1	$\frac{(m-1)^{-1}(\sum_i S_i)^{-1}}{(m-1)^{-1}(\sum_i S_i)^{-1}} = 1$			
(M_6) locally shift t_i	P_{st}	1	$\frac{(2d+1)^{-1}}{(2d+1)^{-1}} = 1$			UNUT
(M_7) merge Δg_i and Δg_{i+1}	P_m	$\frac{P_s}{P_m}$	$\frac{(m-1)^{-1}2(S_i+S_{i+1})^{-1}\binom{S_i+S_{i+1}}{S_i}^{-1}}{(m-1)^{-1}} = \frac{2}{(S_i+S_{i+1})\binom{S_i+S_{i+1}}{S_i}}$			
(M_8) split Δg_i	P_s	$\frac{P_m}{P_s}$	$\frac{(m-1)^{-1}}{(m-1)^{-1}(S_i/2)^{-1}\binom{S_i}{x}^{-1}} = (S_i/2)\binom{S_i}{x}$			
(M_9) create new Δg_i	P_{ag}	$\frac{P_{dg}}{P_{ag}}$	$\frac{(m+1)^{-1}}{(N-m)^{-1}n^{-2}} = \frac{(N-m)n^2}{m+1}$			
(M_{10}) delete Δg_i	P_{dg}	$\frac{P_{ag}}{P_{dg}}$	$\frac{(N-m-1)^{-1}n^{-2}}{m^{-1}} = \frac{m}{(N-m-1)n^2}$			

Table 1: The move sets under different settings. E_1 is the total number of edges in G_1, p_{max} is the maximum parent set size, s_{max} is the maximum number of edge changes allowed in a single transition time, and s_i is the number of edge changes in the set Δg_i. The proposal ratio is the product of the last two columns. The KNKT setting uses moves M_1–M_5, KNUT uses moves M_1–M_6, and UNUT uses moves M_1–M_{10}, in each case with the proposal probabilities appropriately normalized to add to 1.

5. Results on Simulated and Real Data Sets

Here we examine both the speed and accuracy of our sampling algorithm under all three settings and on both simulated and real data sets. We want to solve real-world problems, but accurate ground truths are often not available to assess performance; therefore, we must rely on simulation studies to provide representative performance estimates for the real problems of interest.

We have studied the performance characteristics of our algorithm in simulation studies that vary by several orders of magnitude in number of observations, number of variables, number of epochs, and network density. In each case, we perform simulations with multiple data sets multiple times with multiple chains to help ensure our results are robust to simulation artifacts. For brevity, we only present a few simulation results here; the broader set of experiments we have conducted yield similar results.

All experiments were run on a 3.6GHz dual-core Intel Xeon machine with 4 GB of RAM.

5.1 Small Simulated Data Set

To evaluate the effectiveness of our method, we first apply it to a small, simulated data set. The first experiment is on a simulated ten node network with 1020 observations and six single-edge changes between seven epochs, where the length of each epoch varies between 20 and 400 observations. The

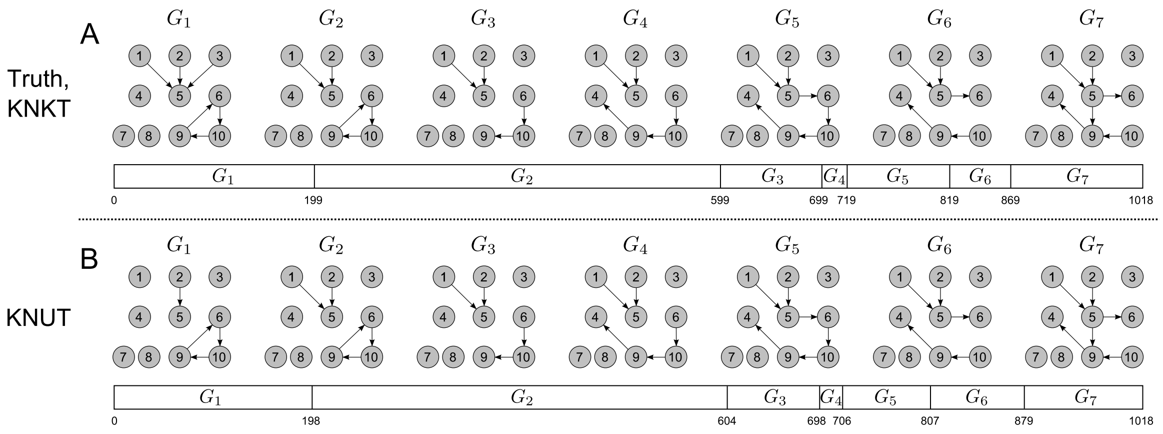


Figure 3: nsDBN structure learning with known numbers of transitions. *A*: True non-stationary data-generation process. The structure evolves gradually from the network at the left labeled G_1 to the network at the right labeled G_7. The epochs in which the various networks are active are shown in the horizontal bars, roughly to scale. The horizontal bars represent the segmentation of the 1020 observations, with the transition times labeled below. When these times are known to the algorithm (the KNKT setting), the recovered nsDBN structure is exactly the true structure. *B*: When the times of the transitions are not known (the KNUT setting), the algorithm learns the model-averaged nsDBN structure shown (selecting edges that occur in greater than fifty percent of the sampled structures). The learned networks and most likely transition times are highly accurate (only missing two edges in G_1 and all predicted transition times close to the truth).

true structure is shown in Figure 3A. We chose a small network with features biologically relevant to genetic regulatory networks: a feedback loop, a variable with at least three parents, a pathway of length six, and the inclusion of observed variables that do not even participate in the network.

5.1.1 KNKT SETTING

In this simple setting, the sampler rapidly converges to the correct solution. We generate a data set using the structure in Figure 3A, and run our sampler for 100,000 iterations, with the first 25,000 samples thrown out for burn-in.

To obtain a consensus (model averaged) structure prediction, an edge is considered present at a particular time if the posterior probability of the edge is greater than 0.5. The value of λ_m has no effect in this setting, and the value of λ_s is varied between 0.1 and 50. The predicted structure is exactly identical to the true structure shown in Figure 3A for a broad range of values, $0.5 \leq \lambda_s \leq 10$, indicating robust and accurate learning.

5.1.2 KNUT SETTING

In this setting, transition times are unknown and must be estimated *a posteriori*. The prior on m remains unused, and for the prior on s, the value of λ_s is again varied between 0.1 and 50.

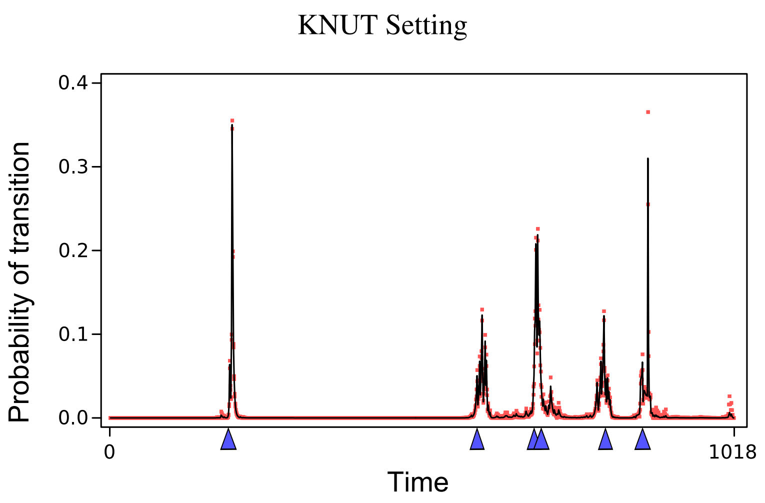


Figure 4: Posterior probability of transition times when learning an nsDBN in the KNUT setting. The blue triangles on the baseline represent the true transition times and the red dots represent one standard deviation from the mean probability, which is drawn as a black line. The variance estimates were obtained using multiple chains. The highly probable transition times correspond closely with the true transition times.

Again, we generate a data set using the structure from Figure 3A and run the sampler for 200,000 iterations, with the first 50,000 samples thrown out for burn-in. More samples are collected in the KNUT setting because we expect that convergence will be slower given the larger space of nsDBNs to explore.

The predicted consensus structure is shown in Figure 3B for $\lambda_s = 5$; this choice of λ_s provides the most accurate predictions. The estimated structure and transition times are very close to the truth. All edges are correct, with the exception of two missing edges in G_1, and the predicted transition times are all within 10 of the true transition times. We can also examine the posterior probabilities of transition times over all sampled structures. This is shown in Figure 4. The blue triangles on the baseline represent the true transition times, and spikes represent transition times that frequently occurred in the sampled structures. While the highest probability regions do occur near the true transition times, some uncertainty exists about the exact locations of t_3 and t_4 since the fourth epoch is exceedingly short.

To obtain more accurate measures of the posterior quantities of interest (such as the locations of transition times), we generate samples from multiple chains; we use 25 in the KNUT setting. Combining the samples from several chains allows us to estimate both the probability of a transition occurring at a certain time and the variance of that estimate. The red dots in Figure 4 represent one standard deviation above and below the estimated mean probability of a transition occurring at a particular time. We discovered that the speed of convergence under the KNKT and KNUT settings were very similar for a given m. This unexpected result implies that the posterior over transition times is rather smooth; therefore, the mixing rate is not greatly affected when sampling transition times.

5.1.3 UNUT SETTING

Finally, we consider the UNUT setting where the number and times of transitions are both unknown. We examine the accuracy of our method in this setting using several values of λ_s and λ_m. We use the range $1 \leq \lambda_s \leq 5$ because we know from the previous settings that the most accurate solutions were obtained using a prior within this range; the range $1 \leq \lambda_m \leq 50$ is selected to provide a wide range of estimates for the prior on m since we have no previous knowledge of what it should be.

Again, we generate a data set using the structure from Figure 3A and run the sampler for 300,000 iterations, with the first 75,000 samples thrown out for burn-in. We collect samples from 25 chains in this setting.

Figure 5 shows the posterior probabilities of transition times for various settings of λ_s and λ_m. As expected, when λ_m increases, the number of peaks decreases. Essentially, when λ_m is large, only the few transition times that *best* characterize the non-stationary behavior of the data will be identified. On the other hand, when λ_m is very small, noises within the data begin to be identified as transition times, leading to poor estimates of transition times.

We can also examine the posterior on the number of epochs, as shown in Figure 6. The largest peak can be used to provide an estimate of m. A smaller λ_m results in more predicted epochs and less confidence about the most probable value of m.

Finally, since we know what the true structure is, we can obtain a precision-recall curve for each value of λ_s and λ_m. The precision-recall curves are shown at the top of Figure 7. To calculate these values, we obtained individual precision and recall estimates for each network at each observation and averaged them over all observations. Therefore, the reported precision and recall values can be viewed as the average precision and average recall over all observations.

One way to identify the best parameter settings for λ_s and λ_m is to examine the best F1-measure (the harmonic mean of the precision and recall) for each. The table in Figure 7 shows the best F1-measures and reveals $\lambda_s = 5$ and $\lambda_m = 1$ as best for this data, although nearly all choices achieve an F1-measure above 0.9.

5.2 Larger Simulated Data Set

To evaluate the scalability of our technique in the most difficult UNUT setting, we also simulate data from a 100 variable network with an average of 50 edges over five epochs spanning 4800 observations, with one to three edges changing between each epoch. We generate 10 different data sets from the model and acquire 25 chains from each data set. For each chain, we take 800,000 samples, with the first 200,000 samples thrown out for burn-in.

The posterior probabilities of transition times and the number of epochs (corresponding to Figures 5 and 6) for one of the simulated data sets are shown in Figure 8. The significantly sharper prediction for the posterior probabilities of transitions occurring at specific times is most likely due to having more observations and, thus, more confident estimates. The number of epochs with the highest posterior probability is five for all choices of priors, which is exactly the true number of epochs for this data set.

Additionally, the precision-recall curves and F1-measures are shown in Figure 9, revealing the $\lambda_s = 1$ and $\lambda_m = 5$ assignments to be best for this data, although all choices achieve excellent results.

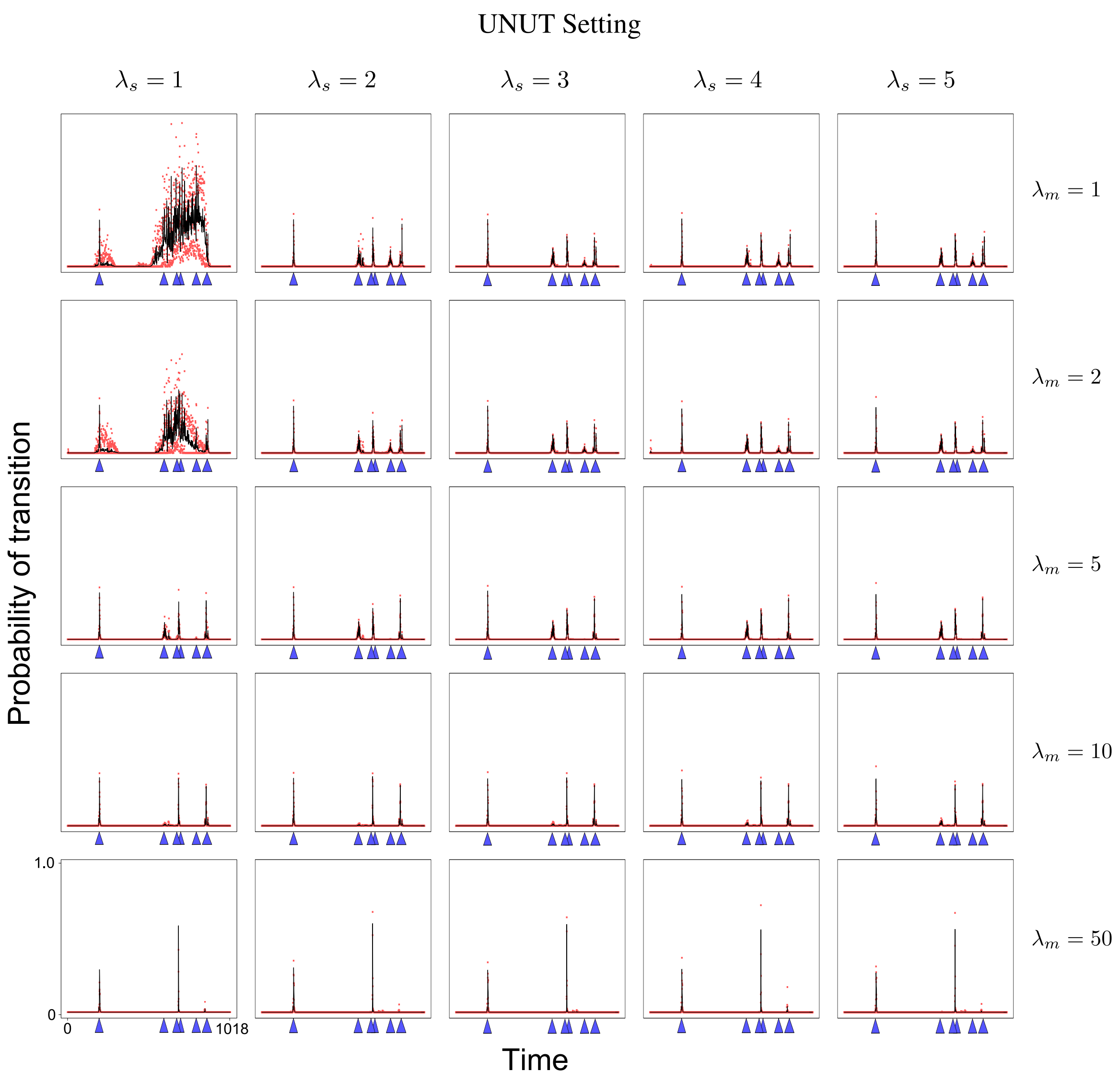


Figure 5: The posterior probabilities of transition times from the sampled structures in the UNUT setting for various values of λ_s and λ_m. As in Figure 4, the blue triangles on the baseline represent the true transition times and the red dots represent one standard deviation from the mean probability obtained from several runs, which is drawn as a black line. Only the (λ_s, λ_m) values of $(1,1)$ and $(1,2)$ seem to result in poor estimates of the true transition times.

5.3 *Drosophila* Muscle Development Gene Regulatory Networks

Having achieved excellent results even in the hardest UNUT setting across ten different data sets generated from two different simulated networks—one with 10 variables and the other with 100—we are confident enough in the usefulness of our model to analyze some real data.

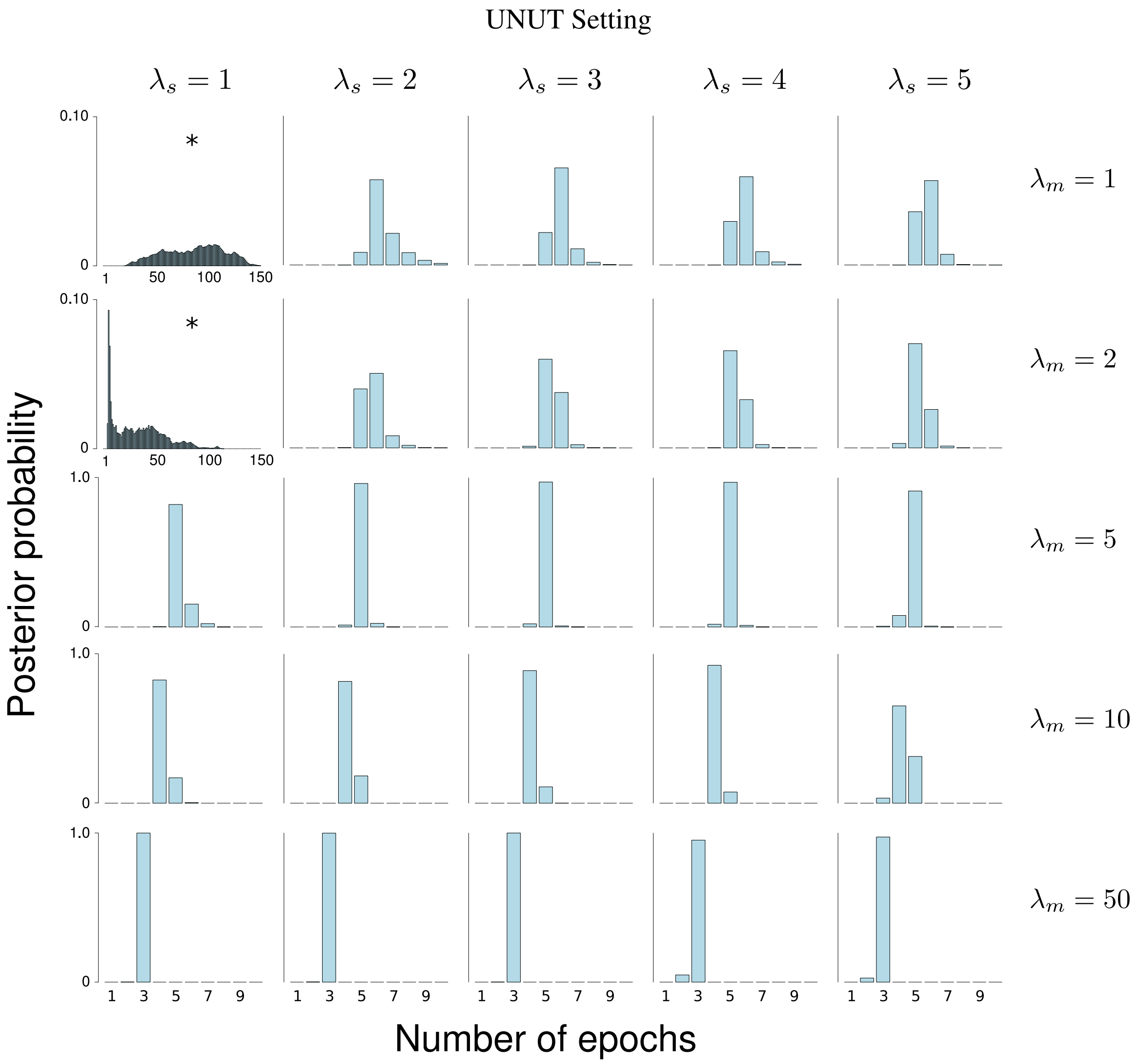
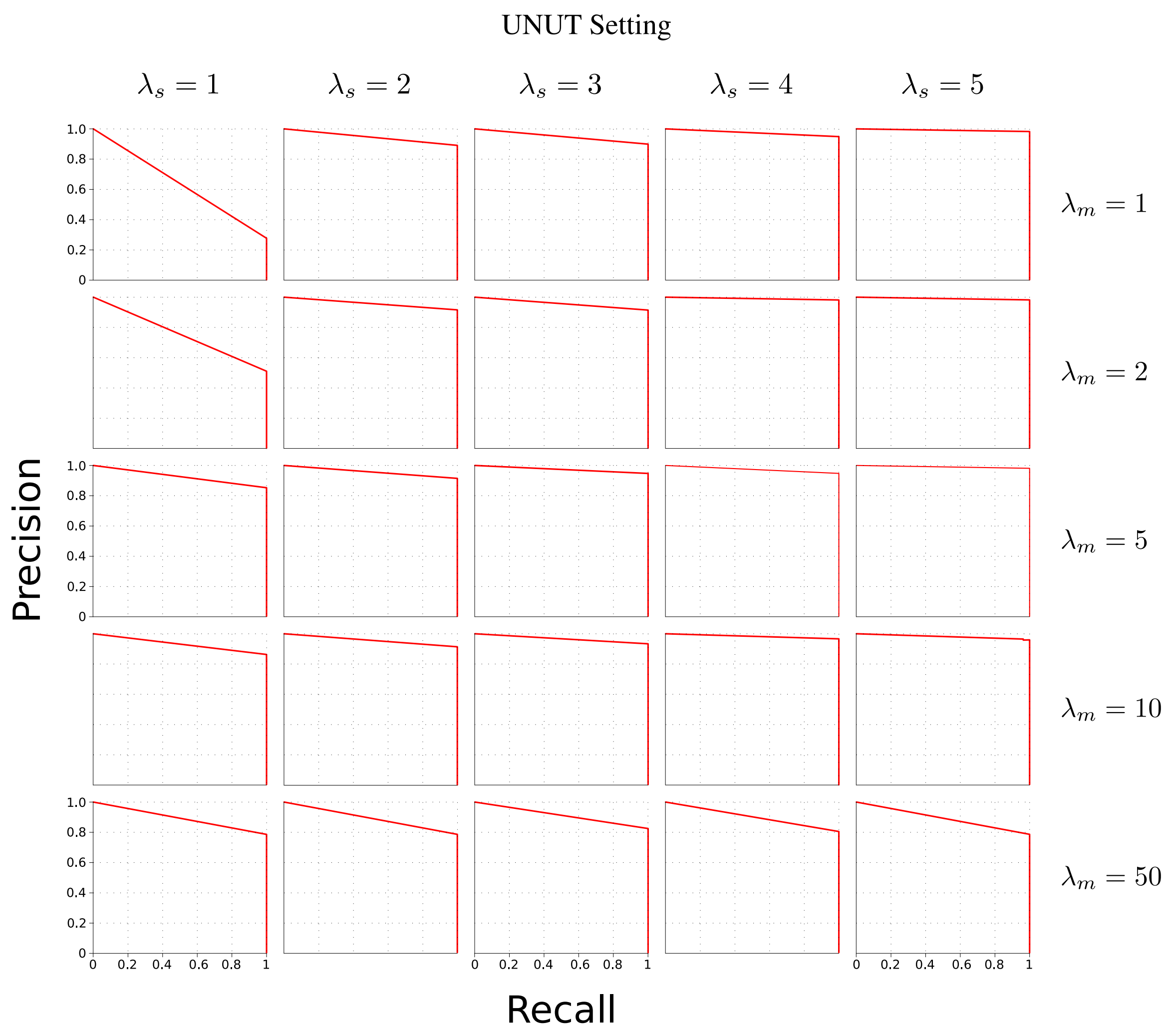


Figure 6: The posterior probabilities of the number of epochs for various values of λ_s and λ_m. The x-axes range from 1 to 10 and the y-axes from 0 to 1 for all of the plots except those marked with a star. The starred plots show predictions with significantly more epochs than the truth. The posterior estimates on the number of epochs m are closest to the true value of 7 when λ_m is 1 or 2.

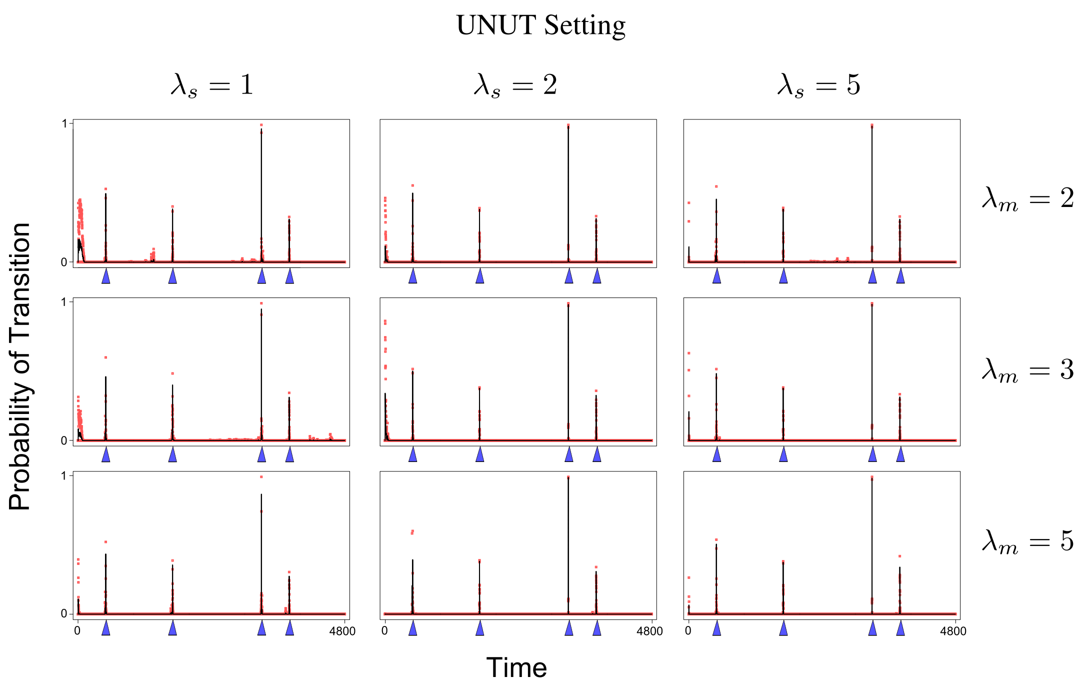
In this subsection, we apply our method to identify non-stationary networks using *Drosophila* development gene expression data from Arbeitman et al. (2002). This data contains expression measurements of 4028 *Drosophila* genes over 66 time steps throughout development and growth during the embryonic, larval, pupal, and adult stages of life. Zhao et al. (2006) focused on 19 genes involved in muscle development and learned a single network over all 66 time steps with this data. Using the same data, Guo et al. (2007) learned a time-varying undirected network over a subset of



λ_s

1	2	3	4	5		
0.4341	0.9423	0.9469	0.9738	**0.9912**	1	
0.6760	0.9562	0.9553	0.9906	0.9909	2	
0.9206	0.9553	0.9729	0.9731	0.9905	5	λ_m
0.9264	0.9550	0.9657	0.9829	0.9791	10	
0.8804	0.8806	0.9042	0.8922	0.8807	50	

Figure 7: *Top*: Precision-recall curves for various values of λ_s and λ_m under the UNUT setting. The most accurate estimates for the structure of the nsDBN arise when $\lambda_s = 5$ and $\lambda_m = 1$. *Bottom*: Corresponding F1-measures for the precision-recall curves. F1-measures over 0.9 are shaded; darker shades indicate values closer to 1, and the highest value is shown in bold.



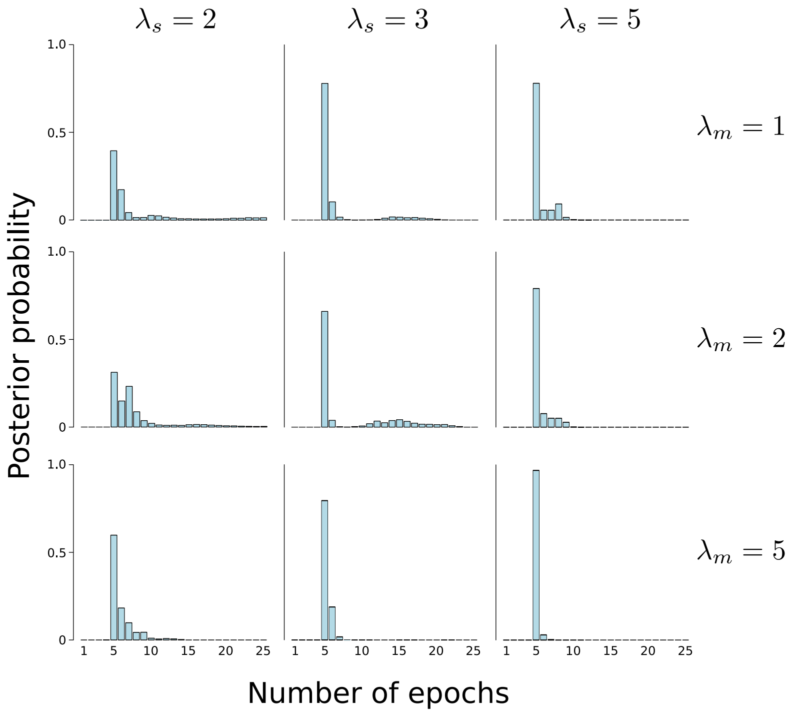
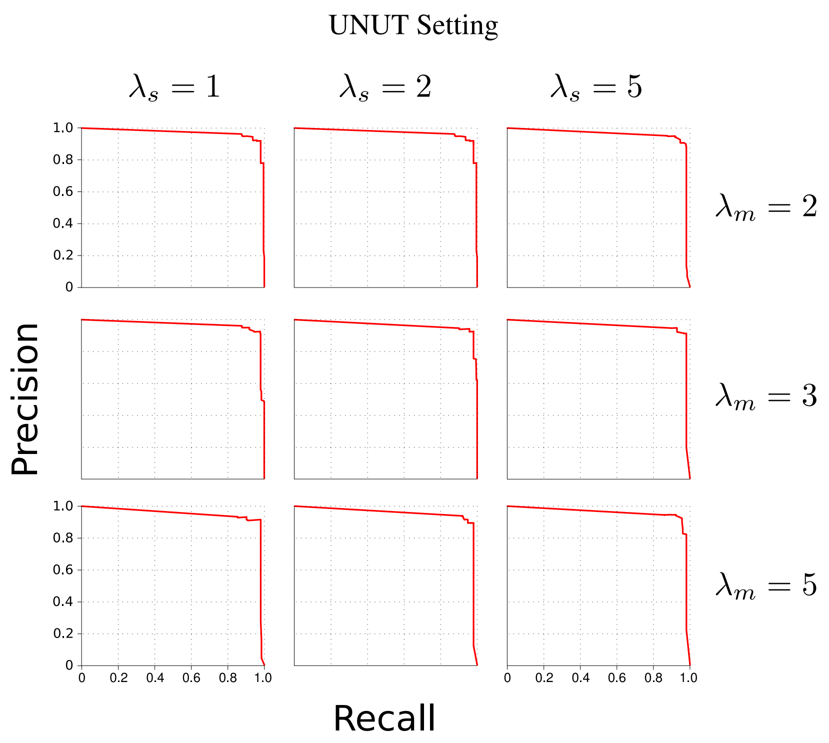


Figure 8: The posterior probabilities of transition times and number of epochs from one of the larger (100 variables, 5 epochs, and 4800 observations) simulated data sets under the UNUT setting for various values of λ_s and λ_m. The axes are the same for all plots. *Top*: The blue triangles on the baseline represent the true transition times and the red dots represent one standard deviation from the mean probability obtained from several runs, which is drawn as a black line.



λ_s

1	3	5		
0.9489	0.9510	0.9468	1	
0.9377	0.9521	0.9356	2	λ_m
0.9531	0.9459	0.9398	5	

Figure 9: *Top*: Precision-recall curves for several values of λ_s and λ_m under the larger 100 variable simulation. *Bottom*: Corresponding F1-measures for the precision-recall curves. F1-measures over 0.9 are shaded; the highest value is shown in bold.

11 of the 19 genes identified by Zhao et al. (2006). To facilitate comparison with as many existing methods as possible, we apply our method to the data describing the expression of the same 11 genes, preprocessing the data in the same way as described by Zhao et al. (2006). Unfortunately, no other techniques predict non-stationary directed networks, so our comparisons are made against the stationary directed network predicted by Zhao et al. (2006) and the non-stationary undirected network predicted by Guo et al. (2007).

We collect 50,000 samples and throw out the first 10,000 for burn-in; we then repeat this process for 25 chains. We need fewer samples in this problem compared to previous data sets because there are relatively few variables and observations.

Figure 10 shows how our predicted structure compares to those reported by Zhao et al. (2006) and Guo et al. (2007). The nsDBN in Figure 10C was learned using the KNKT setting with transition times defined at the borders between the embryonic, larval, pupal, and adult stages.

While all three predictions share many edges, certain similarities between our prediction and one or both of the other two predictions are of special interest. In all three predictions, a cluster seems to form around *myo61f*, *msp-300*, *up*, *mhc*, *prm*, and *mlc1*. All of these genes except *up* are in the myosin family, which contains genes involved in muscle contraction. Within the directed predictions, *msp-300* primarily serves as a hub gene that regulates the other myosin family genes. It is interesting to note that the undirected method predicts connections between *mlc1*, *prm*, and *mhc* while neither directed method makes these predictions. Since *msp-300* seems to serve as a regulator to these genes, the method of Guo et al. (2007) may be unable to distinguish between direct and indirect interactions, due to its undirected nature and reliance on correlations.

Two interesting temporal similarities arise when comparing our predictions to those from Guo et al. (2007). First, an interaction between *eve* and *actn* arise at the beginning of the pupal stage in both methods. Second, the connection between *msp-300* and *up* is lost in the adult network. Note that the loss of this edge actually characterizes the progression to the adult stage from the pupal stage in our prediction, while the method from Guo et al. (2007) combines the two stages. The estimation of a combined pupal/adult stage may simply be due to predicting the loss of the edge between *msp-300* and *up* earlier in development than our method.

Despite the similarities, some notable differences exist between our prediction and the other two predictions. First, we predict interactions from *myo61f* to both *prm* and *up*, neither of which is predicted in the other methods, suggesting a greater role for *myo61f* during muscle development. Also, we do not predict any interactions with *twi*. During muscle development in *Drosophila*, *twi* acts as a regulator of *mef2* which in turn regulates some myosin family genes, including *mlc1* and *mhc* (Sandmann et al., 2006; Elgar et al., 2008); our prediction of no connection to *twi* mirrors this biological behavior. Finally, we note that in our predicted structure, *actn* never connects as a regulator (parent) to any other genes, unlike in the network predicted by Zhao et al. (2006). Since *actn* (actinin) only binds actin, we do not expect it to regulate other muscle development genes, even indirectly.

If we transition to the UNUT setting, we can also examine the posterior probabilities of transition times and epochs. These plots are shown in Figure 11A and 11B, respectively. The transition times with high posterior probabilities correspond well to the embryonic$\rightarrow$larval and the larval$\rightarrow$pupal transitions, but a posterior peak occurs well before the supposed time of the pupal$\rightarrow$adult transition; this reveals that the gene expression program governing the transition to adult morphology is active well before the fly emerges from the pupa, as would clearly be expected. Also, we see that the most probable number of epochs is three to four, mirroring closely the total number of developmental stages.

5.4 Simulated Data Set Similar to the *Drosophila* Data Set

To evaluate the accuracy of a recovered nsDBN on a problem of exactly the same size as the predicted *Drosophila* muscle development network, we simulate a non-stationary time-series with the same number of nodes and a similar level of connectivity as the *Drosophila* data set. We generate data from an nsDBN with 66 observations and transition times at 30, 40, and 58 to mirror the number of observations in embryonic, larval, pupal, and adult stages of the experimental fly data. Since

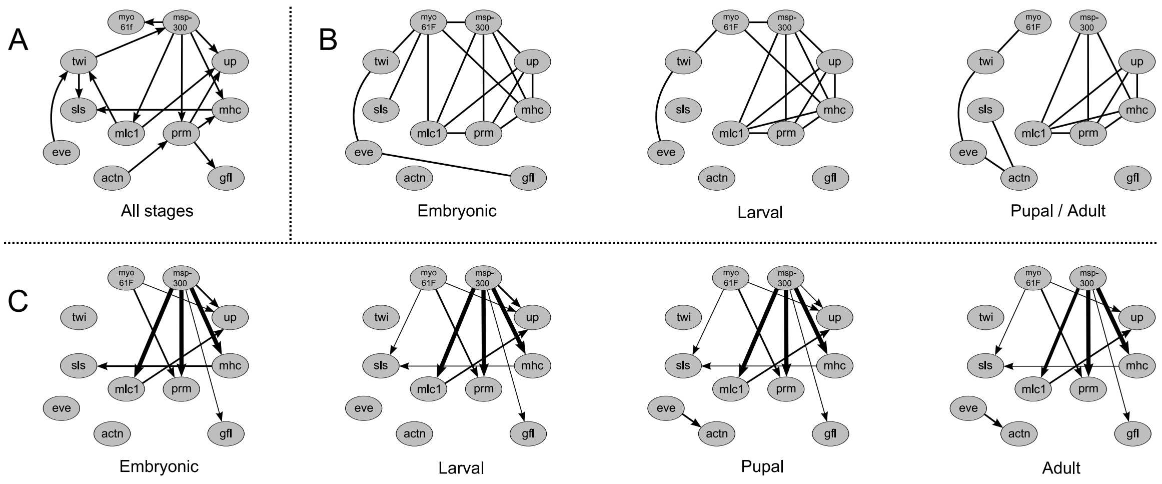


Figure 10: Comparison of computationally predicted *Drosophila* muscle development networks. *A*: The directed network reported by Zhao et al. (2006). *B*: The undirected networks reported by Guo et al. (2007). *C*: The nsDBN structure learned under the KNKT setting with $\lambda_s = 2$. Only the edges that occurred in greater than 50 percent of the samples are shown, with thicker edges representing connections that occurred more frequently.

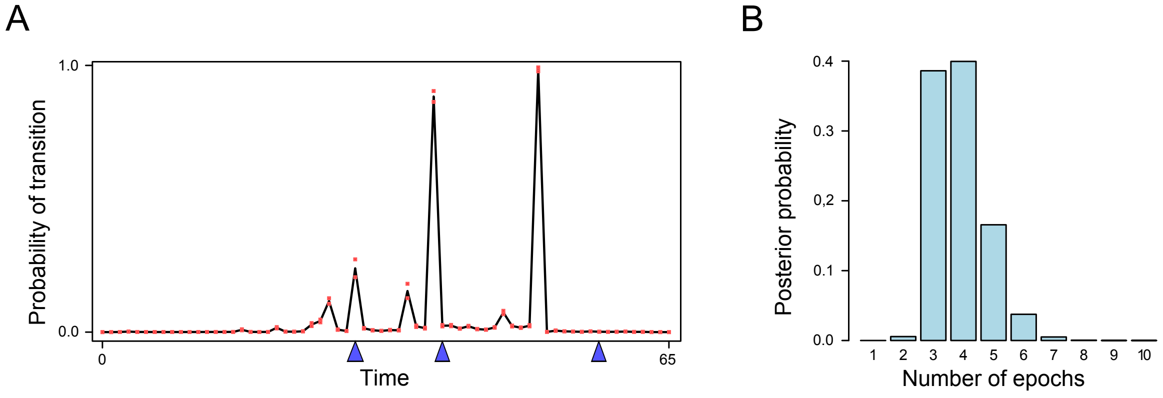
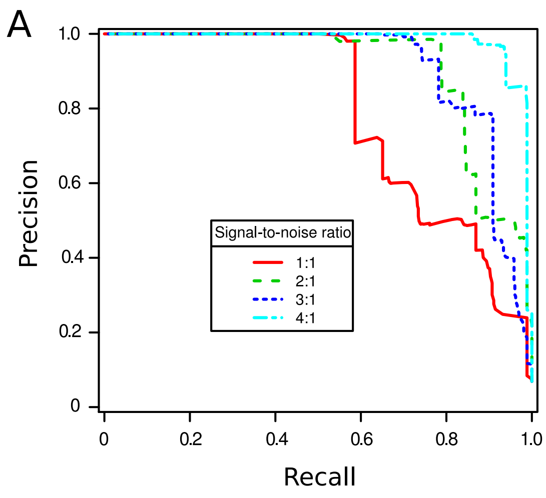


Figure 11: Learning nsDBN structure in the UNUT setting using the *Drosophila* muscle development data. *A*: Posterior probabilities of transition times using $\lambda_m = \lambda_s = 2$. The blue triangles on the baseline represent the borders of embryonic, larval, pupal, and adult stages. *B*: Posterior probability of the number of epochs. The high weight for 3 and 4 epochs closely matches the true number of developmental stages.

it is difficult to estimate the amount of noise in the experimental data, we also simulate noise at various signal-to-noise ratios, from 4:1 down to 1:1. Finally, since many biological processes have more variables than observations, we examine the effect of increasing the number of experimental replicates as a possible means to overcome this challenge.



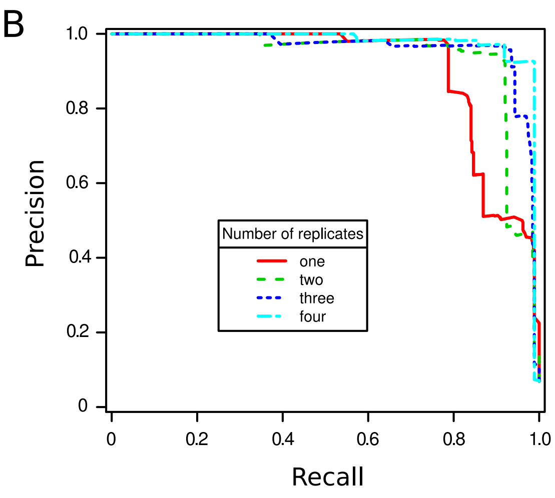


Figure 12: An nsDBN was learned on simulated data that mimicked the number of nodes, connectivity, and transition behavior of the experimental fly data. This allowed us to estimate the accuracy of learned nsDBNs on a problem of this size. *A*: Precision-recall curves for increasing values of the signal to noise ratio in the data (using one replicate). *B*: Precision recall curves for an increasing number of experimental replicates (using an SNR of 2:1). A greater signal to noise ratio and a greater number of experimental replicates both result in better performance, as expected.

As discussed earlier, to obtain posterior estimates of quantities of interest, such as the number of epochs or transition times, we generate many samples from several chains; averaging over chains provides a more efficient exploration of the sample space. To incorporate replicates into the posterior calculations, we generate samples from multiple chains (25) for each set of replicates. Since the underlying data generation process is the same for each replicate, we simply average over all the chains. The results of these simulations are summarized in Figure 12.

As expected, as the signal-to-noise ratio of the data increases, the greater the accuracy in the learned nsDBNs as reflected in the F1-measures: 1:1 is 0.734, 2:1 is 0.850, 3:1 is 0.875, and 4:1 is 0.950. Additionally, increasing the number of replicates also increases prediction accuracy: one is 0.869, two is 0.924, three is 0.945, and four is 0.956. This demonstrates the importance of multiple replicates for biological data with many variables but few observations.

This simulation study allows us to explore how much a relatively small data set and noise affect the predictions from our algorithm on a problem of similar size as the *Drosophila* muscle development network. From the results in Figure 12, we see that learning an nsDBN on a problem of the same scale as the *Drosophila* data set results in accurate network reconstruction, even in the presence of substantial noise. Therefore, we can surmise that any inaccuracies in our predicted *Drosophila* muscle development network would not arise due to the use of a dataset of that size, but might arise from an exceptionally high level of noise in the data (or inappropriate modeling assumptions).

5.5 Neural Information Flow Networks in Songbirds

Our goal is to learn neural information flow networks in the songbird brain. Such networks represent the transmission of information between different regions of the brain. Like roads, the anatomical connectivity of a brain indicates potential pathways along which information can travel. Like traffic, neural information flow networks represent the dynamic utilization of these pathways. By identifying the neural information flow networks in songbirds during auditory stimuli, we hope to understand how sounds are stored and processed in the brain.

In this experiment, eight electrodes were placed into the vocal nuclei of six female zebra finches. Voltage changes were recorded from populations of neurons while the birds were provided with four different two-second auditory stimuli, each presented twenty times. The resulting voltages were post-processed with an RMS transformation and binned to 5 ms; this interval was chosen because it takes 5–10 ms for a neural signal to propagate through one synaptic connection (Kimpo et al., 2003). We analyze data recorded from electrodes for two seconds pre-stimulus, two seconds during stimulus, and two seconds post-stimulus. We learn an nsDBN for two of the birds over six seconds for two different stimuli using all repetitions; this data set contains 8 variables and nearly 25,000 observations for each bird and each stimulus.

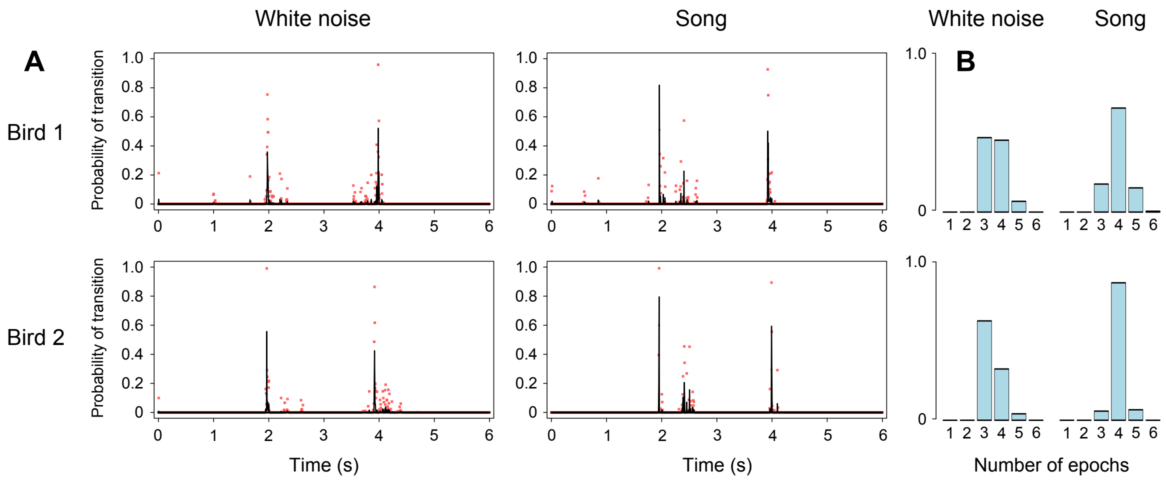


Figure 13: Posterior results of learning nsDBNs under the UNUT setting for two birds presented with two different stimuli (white noise and song). *A*: Posterior transition time probabilities. Transitions are consistently predicted near the stimulus onset (2 seconds) and offset (4 seconds). *B*: Posterior estimates of the number of epochs. The estimated number of epochs is three or four, with strong support for the value four when the bird is presented with a song.

The posterior transition time probabilities and the posterior number of epochs for two birds presented with two different stimuli under the UNUT setting ($\lambda_s = \lambda_m = 2$) can be seen in Figure 13. Note how the estimated transition times correspond closely with the times of the stimulus onset and offset and how the posterior estimate of the number of epochs is around three or four; taken together, these statistics imply that different networks predominate in the pre-stimulus, during stimulus, and post-stimulus time periods.

The posterior estimates consistently differ when the bird is listening to white noise versus song. When listening to a song, an additional transition is predicted 300–400 ms after the onset of a song but not after the onset of white noise. This implies that the bird further analyzes a sound after recognizing it (e.g., hearing a known song), but performs no further analysis when it does not recognize a sound (e.g., hearing white noise).

Previous analysis of this data assumed that any changes in the neural information flow network of a songbird listening to sound occurred only at sound onset and offset (Smith et al., 2006). Only by appropriately modeling the neural information flow networks as nsDBNs are we able to learn that this assumption is not accurate.

Further analysis and investigation of this data is left to future work.

5.6 Performance and Scalability

Due to the use of efficient data structures in the sampler implementation, the computational time needed to update the likelihood is essentially the same for all moves. Therefore, the runtimes of the algorithm under the KNKT, KNUT, and UNUT settings do not differ for a given number of samples. Nevertheless, one typically wants more samples in settings with increased uncertainty to ensure proper convergence.

For the small ten variable simulated data set, the sample collection process for each chain takes about 10 seconds per 100,000 samples, which translates to 10 seconds for the KNKT setting, 20 seconds for the KNUT setting, and 30 seconds for the UNUT setting. Fortunately, all runs can easily be executed in parallel. Sample collection for the *Drosophila* data set and the similarly sized simulated data set takes only a few seconds for each chain under all settings.

For the larger 100 variable simulated data set, sample collection takes about 2 minutes per 100,000 samples. The increased runtime is primarily due to the larger number of variables, so defining the neighborhood for each move takes more time. Due to intelligent caching schemes, the number of observations affects runtime in only a sublinear fashion (provided that enough memory is available).

Surprisingly, one of the largest contributors to running time is the actual recording of the MCMC samples. For example, each sample in the larger simulated data set can be represented by a 10,000 by 4,800 binary matrix of indicators for individual edges at every point in time. A full recording of each sample is therefore very time consuming: just recording each sample in the larger simulated data set leads to an increased runtime of about 50 minutes per 100,000 samples. We can alleviate this problem in several ways. First, because only a small number of those 48 million values actually change between samples, each sample can be represented and output in a compressed fashion; however, the same amount of processing still must occur after the sample collection completes. A better option is to only record the posterior quantities of interest. For example, recording just the transition times and number of epochs adds only a few seconds to the runtime on the larger simulated data set.

6. Discussion

Non-stationary dynamic Bayesian networks provide a useful framework for learning Bayesian networks when the generating processes are non-stationary. Using the move sets described in this paper, nsDBN learning is efficient even for networks of 100 variables, is generalizable to situations of varying uncertainty (KNKT, KNUT, and UNUT), and is robust (i.e., not very sensitive) to the choice of hyper-parameters over a large range of values. Additionally, by using a sampling-based approach, our method allows us to assess a confidence for each predicted edge—an advantage that neither Zhao et al. (2006) nor Guo et al. (2007) share.

We have demonstrated the feasibility of learning an nsDBN in all three settings using simulated data sets of various numbers of transition times, observations, variables, epochs, and connection densities. Additionally, we have identified nsDBNs in the KNKT and UNUT settings using biological gene expression data. The *Drosophila* muscle development network we predict is consistent with the predictions from other techniques and conforms to many known biological interactions from the literature. The predicted transition times and number of epochs also correspond to the known times of large developmental changes. Although each connection on the predicted *Drosophila* muscle development network is difficult to verify, simulated experiments of a similar scale demonstrate highly accurate predictions, even with moderately noisy data and one replicate.

While we focus on certain aspects of the model in this paper, many of our decisions are choices rather than restrictions. For example, we present results using a Markov lag of one, but any Markov lag could be used. Additionally, we use the BDe score metric, but any score metric or conditional independence test can be used instead; however, any score metric which does not integrate over the non-structural parameters would require an augmented sampling procedure. The assumption of discrete data is not necessary; our method easily extends to continuous data, provided that an appropriate scoring metric (like BG) is adapted.

A discrete view of time is necessary to our approach, but many continuous-time data sets can be transformed into discrete-time ones without significant loss of information. The use of directed graphs is also necessary, and desired, but undirected estimates can be obtained through moralization of directed estimates. Although we choose simple priors to learn smoothly evolving networks, nearly any priors would be easy to incorporate; in particular, incorporating expert knowledge about the problem domain would be an ideal method for defining priors.

For problems of more than a few variables, the use of MCMC sampling is essential since EM techniques would not converge in any reasonable time frame given such a large sample space. One of our key discoveries for increasing convergence is the reformulation of the problem from learning multiple networks to learning a network and changes to that network. This parameterization provides an intuitive means for defining evolving networks and allows us to define move sets with good convergence properties. Our particular choices of the move sets are not the only possible ones, but we have taken extra care to ensure that they work well on the types of problems we examine in this paper.

The proposed sampling algorithm scales well to problems with hundreds of variables and thousands of observations, but we are not certain how well it will scale to problems that are orders of magnitude larger. When obtaining sample runs takes days instead of minutes or hours, it may become desirable to obtain faster estimates, even if they are approximate. Variational methods are one alternative to MCMC sampling approaches, often obtaining faster estimates at the cost of decreased accuracy (Beal and Ghahramani, 2006). For the scale of problems in this paper, the run times and convergence rates of our MCMC sampling algorithm were good enough that we did not have to resort to variational approximations. However, if we wish to explore much larger data sets in the future, we may need to develop a variational algorithm to obtain results in a reasonable time frame.

Non-stationary DBNs offer all of the advantages of DBNs (identifying directed, potentially non-linear interactions between variables in multivariate time-series) and are additionally able to identify non-stationarities in the interactions between variables. The sampling algorithm presented here allows one to estimate the strength of individual edges as well as posterior distributions of and quantities of interest. In future work, we hope to analyze data from other fields that have traditionally used DBNs and instead use nsDBNs to identify and model previously unknown or uncharacterized non-stationary behavior.

Another direction that could be explored in the future is nsDBN learning with latent variables. Following an EM approach like in Friedman (1997) would be the first step, but one would also need to consider how to connect hidden variables across epochs and how to incorporate different numbers of hidden variables at different epochs. Learning non-stationary networks with latent variables seems to present a vexing challenge in the general case, but might be feasible in simple cases when enough data is available.

Acknowledgments

The authors would like to thank the reviewers for their insightful comments and suggestions. A.J.H. gratefully acknowledges funding for this work from an NSF CAREER award (0347801), an Alfred P. Sloan Fellowship, and a grant from NIH in Collaborative Research in Computational Neuroscience (R01-DC007996-01).

Appendix A.

While we decided to place a (truncated) geometric prior on the number of epochs m, other priors may be considered. Talih and Hengartner (2005) chose to model epoch lengths as i.i.d. geometric random variables. Here, we prove that a geometrically distributed prior on epoch lengths is equivalent to a geometrically distributed prior on the number of epochs.

Letting l_i be the length of epoch i and N be the total number of observations, we can write a geometrically distributed prior on epoch lengths as:

$$\begin{aligned}\prod_{i=1}^{m}(1-p)^{l_i-1}p &= p^m\prod_{i=1}^{m}(1-p)^{l_i-1}\\ &= p^m(1-p)^{\sum_{i=1}^{m} l_i-1}\\ &= p^m(1-p)^{N-m}\\ &= \left(\frac{p}{1-p}\right)^m(1-p)^N.\end{aligned}$$

Since N is the same for every nsDBN, we see that the geometrically distributed prior is simply a function of the number of epochs m. Compare this to a geometrically distributed prior on the number of epochs, shown below:

$$\begin{aligned}(1-q)^{m-1}q &= (1-q)^m\frac{q}{1-q}\\ &= A(1-q)^m.\end{aligned}$$

where A is a constant that is the same for all nsDBNs. Therefore, a geometrically distributed prior on epoch lengths with success probability $p=\frac{1-q}{2-q}<1/2$ is equivalent to a geometrically distributed prior on the number of epochs with success probability q. In this paper, we assume that q takes the form $q=1-\mathrm{e}^{-\lambda}$, allowing for a more intuitive control of the prior by simply changing λ.

References

Amr Ahmed and Eric P. Xing. TESLA: Recovering time-varying networks of dependencies in social and biological studies. *Proceedings of the National Academy of Sciences*, 106(29):11878–11883, 2009.

Michelle N. Arbeitman, Eileen E.M. Furlong, Farhad Imam, Eric Johnson, Brian H. Null, Bruce S. Baker, Mark A. Krasnow, Matthew P. Scott, Ronald W. Davis, and Kevin P. White. Gene expression during the life cycle of *Drosophila melanogaster*. *Science*, 5590(297):2270–2275, Sep 2002.

Matthew J. Beal and Zoubin Ghahramani. Variational Bayesian learning of directed graphical models with hidden variables. *Bayesian Analysis*, 1(4):793–832, 2006.

Allister Bernard and Alexander J. Hartemink. Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. In *Pacific Symposium on Biocomputing*, volume 10, pages 459–470. World Scientific, Jan 2005.

Wray Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

Carlos M. Carvalho and Mike West. Dynamic matrix-variate graphical models. *Bayesian Analysis*, 2(1):69–98, 2007.

David Maxwell Chickering, Dan Geiger, and David Heckerman. Learning Bayesian networks is NP-Hard. Microsoft Research Technical Report MSR-TR-94-17, Microsoft, Nov 1994.

David Maxwell Chickering, Dan Geiger, and David Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*, pages 112–128. Society for Artificial Intelligence in Statistics, Jan 1995.

Lonnie Chrisman. A roadmap to research on Bayesian networks and other decomposable probabilistic models. CMU technical report, School of Computer Science, CMU, May 1998.

Luís Miguel de Campos, Juan M. Fernandez-Luna, José Antonio Gámez, and José Miguel Puerta. Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, 31(3):291–311, Nov 2002.

Stuart J. Elgar, Jun Han, and Michael V. Taylor. *mef2* activity levels differentially affect gene expression during *Drosophila* muscle development. *Proceedings of the National Academy of Sciences*, 105(3):918–923, Jan 2008.

Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the 14th International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann Publishers, 1997.

Nir Friedman and Zohar Yakhini. On the sample complexity of learning Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 274–282. Morgan Kaufmann Publishers Inc., Oct 1996.

Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI98)*, pages 139–147. Morgan Kaufmann Publishers Inc., 1998.

Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian networks to analyze expression data. In *Research in Computational Molecular Biology (RECOMB00)*, volume 4, pages 127–135. ACM Press, Apr 2000.

Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996, Apr 2000.

Paolo Giudici and Robert Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50(1–2), Jan 2003.

Paolo Giudici, Peter Green, and Claudia Tarantola. Efficient model determination for discrete graphical models. Technical report, Athens University of Economics and Business, 1999.

Marco Grzegorczyk, Dirk Husmeier, Kieron D. Edwards, Peter Ghazal, and Andrew J. Millar. Modelling non-stationary gene regulatory processes with a non-homogeneous Bayesian network and the allocation sampler. *Bioinformatics*, 24(18):2071–2078, Jul 2008.

Fan Guo, Wenjie Fu, Yanxin Shi, and Eric P. Xing. Reverse engineering temporally rewiring gene networks. In *NIPS workshop on New Problems and Methods in Computational Biology*, Dec 2006.

Fan Guo, Steve Hanneke, Wenjie Fu, and Eric P. Xing. Recovering temporally rewiring networks: A model-based approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML07)*, Jun 2007.

Steve Hanneke and Eric P. Xing. Discrete temporal models of social networks. In *Workshop on Statistical Network Analysis at the 23rd International Conference on Machine Learning*, Jun 2006.

Alexander J. Hartemink, David K. Gifford, Tommi S. Jaakkola, and Richard A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Pacific Symposium on Biocomputing*, volume 6, pages 422–433. World Scientific, Jan 2001.

David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, Sep 1995.

Reimar Hofmann and Volker Tresp. Discovering structure in continuous variables using Bayesian networks. In *Advances in Neural Information Processing Systems 8 (NIPS95)*, pages 500–506. MIT Press, Dec 1995.

Rhea R. Kimpo, Frederic E. Theunissen, and Allison J. Doupe. Propagation of correlated activity through multiple stages of a neural circuit. *Journal of Neuroscience*, 23(13):5750–5761, 2003.

Mladen Kolar, Le Song, Amr Ahmed, and Eric P. Xing. Estimating time-varying networks. *The Annals of Applied Statistics*, 4(1):94–123, Mar 2010.

Paul K. Krause. Learning probabilistic networks. *The Knowledge Engineering Review*, 13(4):321–351, 1998.

Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(4):269–293, Jul 1994.

Pedro Larrañaga, Mikel Poza, Yosu Yurramendi, Roberto H. Murga, and Cindy M.H. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Journal on Pattern Analysis and Machine Intelligence*, 18(9):912–926, Sep 1996.

Nicholas M. Luscombe, M. Madan Babu, Haiyuan Yu, Michael Snyder, Sarah A. Teichmann, and Mark Gerstein. Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature*, 431:308–312, Sep 2004.

David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International Statistical Review*, 63(2):215–232, Aug 1995.

Dimitris Margaritis. Distribution-free learning of Bayesian network structure in continuous domains. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI05)*, pages 825–830. AAAI Press / The MIT Press, Jul 2005.

Kevin Murphy. Learning Bayesian network structure from sparse data sets. UC Berkeley technical report 990, Computer Science Department, University of California at Berkeley, May 2001.

Thomas Sandmann, Lars J. Jensen, Janus S. Jakobsen, Michal M. Karzynski, Michael P. Eichenlaub, Peer Bork, and Eileen E.M. Furlong. A temporal map of transcription factor activity: *mef2* directly regulates target genes at all stages of muscle development. *Developmental Cell*, 10(6): 797–807, Jun 2006.

V. Anne Smith, Erich D. Jarvis, and Alexander J. Hartemink. Influence of network topology and data collection on network inference. In *Pacific Symposium on Biocomputing*, volume 8, pages 164–175. World Scientific, Jan 2003.

V. Anne Smith, Jing Yu, Tom V. Smulders, Alexander J. Hartemink, and Erich D. Jarvis. Computational inference of neural information flow networks. *PLoS Computational Biology*, 2(11): 1436–1449, Nov 2006.

Joe Suzuki. Learning Bayesian belief networks based on the minimum description length principle: An efficient algorithm using the branch and bound technique. In *Proceedings of the 13th International Conference on Machine Learning (ICML96)*, pages 462–470. Morgan Kaufmann Publishers Inc., Jul 1996.

Makram Talih and Nicolas Hengartner. Structural learning with time-varying components: Tracking the cross-section of financial time series. *Journal of the Royal Statistical Society B*, 67(3):321–341, Jun 2005.

Claudia Tarantola. MCMC model determination for discrete graphical models. *Statistical Modelling*, 4(1):39–61, Apr 2004.

Stanley Wasserman and Philippa E. Pattison. Logit models and logistic regressions for social networks: I. An introduction to Markov graphs and $p*$. *Psychometrika*, 61(3):401–425, Sep 1996.

Xiang Xuan and Kevin Murphy. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th International Conference on Machine Learning (ICML07)*, Jun 2007.

Wentao Zhao, Erchin Serpedin, and Edward R. Dougherty. Inferring gene regulatory networks from time series data using the minimum description length principle. *Bioinformatics*, 22(17): 2129–2135, Sep 2006.