



university of  
 groningen

faculty of science  
 and engineering

# A new Bayesian network model for learning static and dynamic interactions from temporal data

Master's Project Mathematics

January 2021

Student: Moschanthi Korre

First supervisor: Prof.dr. Marco Grzegorzcyk

Second assessor: dr. Wim Krijnen

## **Abstract**

Bayesian networks are a popular modelling tool for learning the conditional (in-)dependencies among variables. If the data set consists of independent (steady state) observations, (static) Bayesian networks can be applied and the relationships are learned in form of a directed acyclic graph. The edges of the graph represent contemporaneous dependencies. From temporal data (time series) directed graphs can be learned with dynamic Bayesian networks. The edges then represent dynamic dependencies; i.e. dependencies that are subject to a time lag. Within this Master project, static and dynamic Bayesian networks are combined into a new mixed Bayesian network model, which can model contemporaneous and dynamic interactions simultaneously from temporal data. Then, the model is applied for the inference of symptom networks, on data collected from patients in successive clinical stages of psychosis.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Methodology</b>	<b>5</b>
2.1	Bayesian network models . . . . .	5
2.1.1	Static Bayesian networks . . . . .	5
2.1.2	Dynamic Bayesian networks . . . . .	10
2.2	Mixed Bayesian networks . . . . .	12
2.3	The Gaussian BGe scoring metric . . . . .	18
2.3.1	Static Bayesian Networks . . . . .	18
2.3.2	Dynamic Bayesian Networks . . . . .	21
2.3.3	Mixed Bayesian networks . . . . .	24
2.4	Markov Chain Monte Carlo . . . . .	27
2.4.1	Motivation . . . . .	27
2.4.2	Mathematical Background . . . . .	28
2.4.3	The MCMC scheme and the Metropolis-Hastings algorithm . . . . .	29
2.4.4	MCMC sampling of Bayesian networks . . . . .	31
2.5	Posterior probability of edge relation features . . . . .	38
2.6	Advanced methods . . . . .	40
2.6.1	The Werhli and Husmeier model . . . . .	40
2.6.2	Introducing node ordering in the Werhli-Husmeier model . . . . .	41
<b>3</b>	<b>Data</b>	<b>48</b>
3.1	The RAF signalling pathway . . . . .	48
3.2	Psychometric data . . . . .	49
<b>4</b>	<b>Implementation details</b>	<b>51</b>
4.1	Synthetic data . . . . .	51
4.1.1	Evaluation of the mixed Bayesian network model . . . . .	51
4.1.2	Comparison between order and structure MCMC . . . . .	52
4.1.3	The coupling scheme . . . . .	52
4.2	Psychometric Data . . . . .	53
<b>5</b>	<b>Results</b>	<b>55</b>
5.1	Evaluation of structure mix BN . . . . .	55
5.2	Order MCMC . . . . .	60
5.3	Integration of data sets derived under different conditions . . . . .	62
5.4	Psychometric data . . . . .	65
<b>6</b>	<b>Discussion-conclusion</b>	<b>70</b>

# 1 Introduction

Bayesian networks([26] ) are a powerful tool of increasing popularity in modern statistical inference, in a wide range of applications -from computational biology to speech and picture processing. The popularity of Bayesian networks stems from the fact that they serve as a compact and natural graphical representation of the joint distribution of the variables in the domain, in the form of a network structure, whose topology encodes important information about the domain.

In the last decades, there has been a great deal of research on the problem of learning Bayesian networks from data. To this end, a scoring function needs to be defined, so that different networks can be evaluated on the given data. A consistent scoring metric in the Bayesian context is the marginal likelihood of a network given the data, which requires the model parameters to be integrated out. The probabilistic models that allow the derivation of a closed form solution for the marginal likelihood is the BGe scoring metric for continuous data [2], and the BDe score for discrete data [33]. The second component for learning a network structure from data is to define a search strategy, that is, a scheme that allows us to explore the space of structures in order to find "high scoring networks". A challenge posed in the framework of Bayesian networks is that the space of structures grows super exponentially to the size of the domain. Heuristic optimization approaches can be used in order to determine an optimum network that maximizes the scoring function. However, these approaches are only useful in case where the data can be adequately explained by one single model, which is not always the case. In many settings, there are more than one structures that serve equally well as evidence for the data. *Markov Chain Monte Carlo* (MCMC) sampling techniques, such as *Structure MCMC* ([28]) and *order MCMC* ([13]) can be employed in such scenarios. These approaches allow us to obtain a sample of networks with high posterior probabilities, which can then be used for edge relation feature estimation, through *Bayesian averaging*.

The concept of Gaussian belief networks was also expanded to the dynamic setting. Dynamic Bayesian networks are graphical probabilistic models which can be employed in case the domain consists of time series data. The edges of a dynamic Bayesian network express interactions that are subject to a time delay, and imply causal influences between variables.

In this thesis, we will combine features of both static and dynamic Bayesian networks, in a new, mix- Bayesian network model, which can learn both contemporaneous and dynamic relations from temporal data. We investigate the performance of the model in comparison to the static and dynamic Bayesian network models. In case there are both static and dynamic underlying dependencies in the domain, we examine whether the new model is able to detect the dominant interactions

and thus provide additional insight compared to the "pure" models. In case there are only static or only dynamic dependencies present in the domain, we want to confirm that the new mixed model performs equally well as the corresponding "pure" models, in detecting the dominant interactions in the domain. The motivating data consists of time series data, and it stems from a recent study on patients in successive stages of psychosis. Our goal is to use the mix Bayesian network model to build symptom networks for each one of these stages. We suspect that dynamic and contemporaneous interactions between symptoms coexist in the spectrum of psychosis, hence employing the new mix model will reveal interesting insights about the data.

We will analyze the methods we used for the analysis in section 2. In subsection 2.2 of this section, we will introduce the new mixed Bayesian network model. Within this section, we will start building up to our inference strategy for the psychometric data, by addressing the problem of learning network structures from data. We will employ the BGe scoring metric as a scoring function. This scoring metric was introduced by Geiger and Heckerman for static Bayesian networks, and thereafter adjusted for the dynamic setting. Under subsection 2.3.3, we show how the BGe score of a mixed Bayesian network can be computed using the same computational steps as in the original case.

We will then move on to describing the search process we will employ, which is based on MCMC inference methods. Two approaches are presented- structure and order MCMC. As we will discuss under subsection 2.4.4, structure MCMC, which was our first approach, comes with significant convergence issues, especially in larger domains. For reasons that will be explained in the same subsection, the mixed BN model can aggravate convergence issues. Order MCMC is proposed as an alternative that we employ to overcome these problems, given that the data that we intend to analyze is rather large.

In the end of the section, we will discuss a coupling scheme that we will employ for the analysis of the psychometric data set, which was proposed by Werhli and Husmeier [24]. This coupling scheme will allow us to infer a different network for every clinical stage, but it contains an intrinsic mechanism that encourages the exchange of information between these networks . In the next subsection (3), we will get into more detail about the data . After giving a brief outline of the experiments we performed in section 4, we present the results of our analysis in section 5 . These results will be discussed under section 6.

## 2 Methodology

In this section, we firstly recapitulate some theoretical background on static and dynamic Bayesian networks in subsections 2.1.1 and 2.1.2, so we can, later on, introduce our new, mixed Bayesian network model in subsection 2.2. In subsection 1.4, we describe the linear Gaussian BGe scoring metric of Geiger and Heckerman [2], [27]. Given a domain  $D$ , this model will allow us to derive a closed form solution for the marginal likelihood of a network  $G$ . Geiger and Heckerman developed the model for pure static observational data, but we will discuss how it has been expanded into the dynamic setting in subsection 2.3.2. In subsection 2.3.3, we will show how we can also expand it to the mixed setting after some straightforward modifications are imposed. Two alternative MCMC approaches for sampling networks from the posterior will be discussed in subsection 2.4.4, *structure MCMC*, as proposed by Madigan and York ([28]), and *order MCMC*, as proposed by Friedman and Koller ([13]). Then, this sample of networks can be used for estimating posterior edge relation features, as we will explain under subsection 2.5. We will conclude the methods section by describing the model proposed by Werhli and Husmeier [24] for the analysis of disjunct data sets obtained under different experimental conditions. The original model, which is based on structure MCMC moves, is described under subsection 2.6.1, and the slightly modified method, which is based on order MCMC moves, is detailed under subsection 2.6.2.

### 2.1 Bayesian network models

In this subsection, we will discuss some basic concepts on the "pure models", that is, the static and dynamic Bayesian network models. Some basic understanding on these models is required, as our new mixed Bayesian network model combines features of both. We will first present *Static Bayesian networks*, in subsection 2.1.1, and, subsequently *Dynamic Bayesian networks*, in 2.1.2. After some necessary background ideas are covered, we can proceed with presenting our new mixed model in the next subsection (2.2).

#### 2.1.1 Static Bayesian networks

Static Bayesian networks ([26]) are probabilistic graphical models that represent the joint distribution of a set of variables  $X_1, X_2, \dots, X_N$  in the form of a *Directed Acyclic Graph* (DAG)  $G$ . The DAG consists of a set of nodes, which correspond to the variables in the domain, and a set of directed edges, which encode direct dependencies between them. A Bayesian network is also described by a set of *local probability distributions* with parameters  $q_i$  associated with the nodes in the graph structure.

The structural features of a Bayesian network reveal important information about the domain. For instance, an edge between two nodes implies a direct correlation between them. The absence of an edge between two nodes means that they are conditionally independent from one another. In other words, if we observe any correlation between them, then this correlation is an indirect one, mediated by other variables.

If there is an edge between two nodes, then we will call these nodes *adjacent*, and *non-adjacent* otherwise. The adjacency relations between the nodes can be represented in matrix form by the *adjacency matrix*, a square binary matrix  $A$ , where  $A_{ij} = 1$  if there is an edge from  $X_i$  to  $X_j$  (symbolically  $X_i \rightarrow X_j$ ), and zero otherwise.

For every node  $X_i$  in  $G$ , we define its parent set  $Pa_G(X_i) = \{X_j, j \in [N] \setminus i | X_j \rightarrow X_i\}$ . We call  $X_j$  an *ancestor* of  $X_i$  if there is a directed path from  $X_j$  to  $X_i$  in  $G$ . In the same context,  $X_i$  is called a *descendant* of  $X_j$ .

The *local Markov assumption* tells us that every node is independent of its non descendants given its parents. Given the local Markov assumption, information on the relations of conditional (in)dependencies between the nodes can be extracted easily using a straightforward criterion called *d-separation*, which allows us to determine whether two nodes  $X_i$  and  $X_j$  are conditionally independent, given a third subset of nodes  $Z$ . For more details on the concept of d-separation, we refer the reader to [26]. The local Markov assumption leads to the following factorization of the joint distribution:

$$P(X_1, X_2, \dots, X_N | G, q) = \prod_{i=1}^N P(X_i | Pa_G(X_i), q_i) \quad (1)$$

where  $q = (q_1, q_1, \dots, q_N)$  is the vector of unknown parameters of the local probability distributions. Therefore, given the topology of a DAG, a unique factorization of the joint distribution is induced. It is, however, possible for more than one DAGs to imply the same (in)dependencies among the variables, and thus yield equivalent factorizations of the joint probability distribution. We then say that these DAGs are *equivalent*. The relation of equivalence partitions the space of DAGs into *equivalence classes*.

A *v-structure* in  $G$  is a subgraph of  $G$ , which consists two non-adjacent nodes that co-parent a third node. It has been proven ([29]) that two DAGs are equivalent if and only if they have the same skeleton and the same *v-structures*. The *skeleton* of a DAG is an undirected structure that

results from the DAG, if we ignore the directions of its edges. Chickering ([31]) has proven that an equivalence class of DAGs can be uniquely represented by its *Completed Partially Directed Acyclic Graph* (CPDAG). A *Completed Partially Directed Acyclic Graph* of a DAG is a graphical object that consists of both directed and undirected edges: An edge which is present in every DAG that belongs in the equivalence class of  $G$ , keeps the direction of the corresponding edge in  $G$  in the CPDAG (*compelled edge*). All edges that participate in v-structures are compelled, and keep their directionality in the CPDAG. However, not all compelled edges participate in a v-structure. More specifically, in a static Bayesian network, an edge is compelled in three cases: If it participates in a v-structure, if its reversal will result in the forming of a v-structure or if its reversal will result in the forming of an invalid cycle.

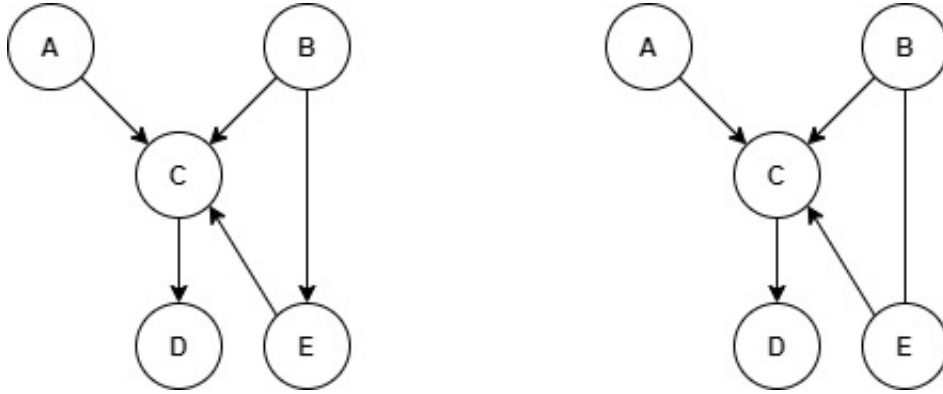


Figure 1: A *Directed Acyclic network* (left panel) and the corresponding *CPAG* (right panel).

If an edge is not compelled, then it is *reversible*, and is represented by an undirected edge in the CPDAG. When an edge is reversible, then there is disagreement between members of the equivalence class on the orientation of this edge.

An example of a DAG and its equivalent CPDAG is illustrated in figure (1). The edge  $A \rightarrow C$  is compelled, as it participates in a v-structure- and the same hold for edges  $B \rightarrow C$ ,  $E \rightarrow C$ . The edge  $C \rightarrow D$  is also compelled, as its reversal will create a v-structure,  $D \rightarrow C \leftarrow A$ . Consequently, any other structure on five nodes, where this edge is reversed, will yield a different factorization of the joint distribution, and thus belong in a different equivalence class. The edge  $B \rightarrow E$  on the other hand, is reversible. This means that there is at least one DAG other than  $G$  in  $G$ 's equivalence class, in which these edges have the opposite direction. In [10], Chickering introduced an algorithm that allows us to convert a DAG into its CPDAG, by classifying the edges of its input DAG argument as either compelled or reversible.

We can now address the problem of learning a network structure from data. Assume our domain consists of  $N$  variables, and we have  $m$  independent realizations for each one of them. The Bayesian paradigm tells that we need to define a prior  $P(G)$  over the space of DAGs on  $N$  nodes. The role of the prior is to integrate, non-data based information into the model. Given a lack of prior knowledge, we can shrink its impact, by choosing a uniform (flat) uninformative prior. However, there are many alternatives in which the prior over networks contributes more drastically in the shape of the posterior distribution.

Once the prior is assessed, it will then be updated according to Bayes rule to yield the posterior distribution of a graph  $G$ :

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)} = \frac{P(D|G)P(G)}{\sum_{G^* \in \mathcal{G}} P(D|G^*) P(G^*)}$$

The *normalization constant*  $P(D)$  is the sum over the space of valid DAGs  $\mathcal{G}$ . Nevertheless, the space of DAGs expands super exponentially as  $N$  increases, which makes this sum intractable, even for relatively small domains. Consequently, and since the normalization constant is independent of  $G$ , we only focus on the numerator  $P(D|G)P(G)$ . The posterior of a structure  $G$  tells us how well the data  $D$  supports structure  $G$ .

The term  $P(D|G)$  is the *marginal likelihood* of the data, given a network  $G$ , and is defined as the integral over the parameter space:

$$P(D|G) = \int P(D, q|G) dq = \int P(D|G, q) P(q|G) dq \quad (2)$$

Where  $P(q|G)$  is the prior distribution of the parameter vector. The marginal likelihood of the data given a network  $G$  tells us how well this network explains the data. The prior over the parameters can be specified according to the model we are employing. In this thesis, we focus on the Gaussian BGe scoring metric of Geiger and Heckerman. For this model, we make sure that this prior satisfies the fairly weak assumptions of parameter independence and parameter modularity (assumptions 4 and 5 in [2]). Parameter independence says that the parameters associated with each node in a Bayesian network are independent. Thus, the prior on the parameter vector  $q$  can be factorized as:

$$P(q|G) = \prod_{i=1}^N P(q_i|G)$$

Parameter modularity claims that the probability of the local parameter vector  $q_i$ , associated with

node  $X_i$  depends only  $Pa_G(X_i)$ . Combining these two assumptions together, we can write:

$$P(q|G) = \prod_{i=1}^N P(q_i|Pa_G(X_i)) \quad (3)$$

Under these two assumptions, a closed form solution of this integral in (2) can be derived ([2]), given that the data is complete. From the local Markov assumption, and for a fixed parameter vector  $q$ , the likelihood term can be factorized as:

$$P(D|G, q) = \prod_{i=1}^N \prod_{j=1}^m P(X_i = D_{i,j} | Pa_G(X_i) = D_{Pa_G(X_i),j}, q_i)$$

where  $D_{i,j}$  is the  $j$ -th realization of  $X_i$ , and  $D_{Pa_G(X_i),j}$  represents the  $j$ -th realisations of the variables in  $Pa_G(X_i)$ . With this in mind, we can derive (ref theorem in Geiger Heckerman):

$$P(D|G) = \prod_{i=1}^N \int P(q_i|Pa_G(X_i)) \prod_{j=1}^m P(X_i = D_{i,j} | Pa_G(X_i) = D_{Pa_G(X_i),j}, q_i) dq_i \quad (4)$$

From now on, we denote  $\pi_i = Pa_G(X_i)$  for convenience. Following the notation of [14], we define:

$$\Psi[D_i^{\pi_i}] = \int P(q_i|\pi_i) \prod_{j=1}^m P(X_i = D_{i,j} | \pi_i = D_{\pi_i,j}, q_i) dq_i \quad (5)$$

with  $D_i^{\pi_i} = \{D_{i,j}, D_{\pi_i,j}, 1 \leq j \leq m\}$  representing the instances of the data that correspond to the realizations of  $X_i$  and its parents  $\pi_i$ .

We can now rewrite the marginal likelihood as a decomposition of *local scores*:

$$P(D|G) = \prod_{i=1}^N \Psi[D_i^{\pi_i}] \quad (6)$$

The Gaussian *BGe* scoring metric of Geiger and Heckerman allows us to compute the local scores analytically in a closed form, by assigning a linear Gaussian distribution to the local conditional distribution  $P(X_i|\pi_i, q_i)$  and choosing the conjugate normal Wishart distribution for the prior distribution of the parameters  $P(q_i|\pi_i)$ . The BGe scoring metric will be thoroughly discussed in the following section.

### 2.1.2 Dynamic Bayesian networks

Dynamic Bayesian networks are graphical models, which we employ in the case where, instead of independent steady-state observations, the data consists of time series data, in the domain  $(X_1(t), X_2(t), \dots, X_n(t))_{t \in T}$ . Here, every directed edge implies a causal effect with a *delay*  $\tau$  between variables, meaning that every realization of a child node is influenced by previous realizations of its parent nodes.

According to the *causal Markov assumption*,  $X_i(t)$  is only influenced by the set  $\{X_j(t - \tau) | X_j \in Pa_G(X_i)\}$ . In other words, the sequence of past events that caused  $Pa_G(X_i)$  have no influence on  $X_i(t)$ . Dynamic Bayesian networks are *homogeneous Markov models*. This means that the transition probabilities between the time slices  $t$  and  $t - \tau$  do not depend on the time point, and thus are equal for all  $t \in T$ .

The interpretation of the directed edges is a core difference between static and dynamic Bayesian networks. In static Bayesian networks, an edge encodes a direct correlation between two nodes. In contrast, an edge in a Dynamic network implies a *causal influence*, and therefore, it cannot imply a bidirectional relation. As a consequence of this, a dynamic Bayesian network induces a unique factorization of the joint distribution. Thus, there are no equivalence classes in dynamic Bayesian networks.

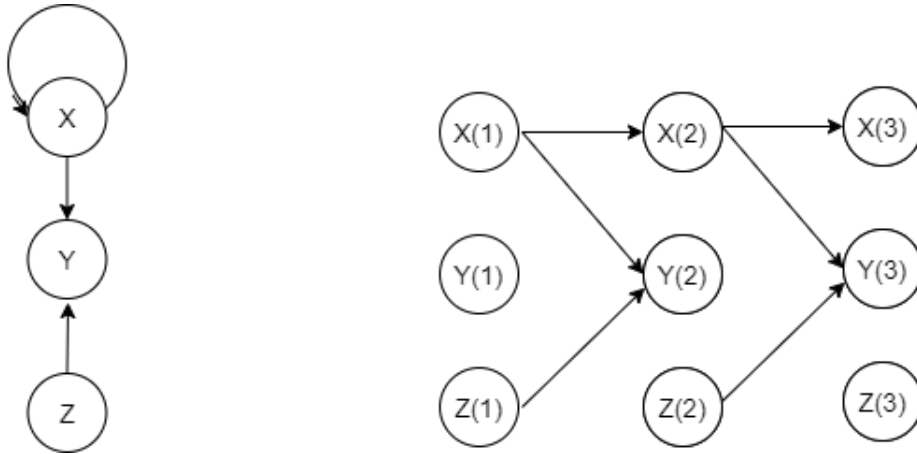


Figure 2: Example of dynamic Bayesian network(left hand side) and the equivalent DAG unfolded in time (right hand side)

Another major difference is that in dynamic Bayesian networks, the acyclicity constraint is lifted. Therefore, a dynamic Bayesian network is defined by a directed (not necessarily acyclic) network, and a family of local probability distributions, with their parameters  $q_i$  corresponding to the nodes

$X_i$ . In figure 2 a dynamic Bayesian network is illustrated. In its succinct representation (left panel) the network consists of three edges, one of which is a directed edge of the form  $X \leftarrow X$ . We call such edges *self loops*. If we unfold the state space in time, we obtain a *directed acyclic network*, as shown in the left panel.

Similar to static Bayesian networks, the marginal likelihood can be decomposed into a product of single-node terms (local scores). Following the notation of [14], we can define:

$$\Psi[D_i^{\pi_i(t-1)}] = \int P(q_i|\pi_i) \prod_{t=2}^m P(X_i = D_{i,t} | \pi_i = D_{\pi_i,t-1}, q_i) dq_i \quad (7)$$

with  $D_i^{\pi_i(t-1)} = \{D_{i,t}, D_{\pi_i,t-1}, 2 \leq t \leq m\}$  representing the instances of the data corresponding to all realizations from node  $X_i$ , from time point  $t = 2$  to time point  $t = m$ , and all realizations of its parents  $\pi_i$ , from time point  $t = 1$  to time point  $t = m - 1$ . The dynamic counterpart for equation 6 is:

$$P(D|G) = \prod_{i=1}^N \Psi[D_i^{\pi_i(t-1)}] \quad (8)$$

Because of the time lag, the effect of the nodes in  $\pi_i$  is not observable at  $t = 1$ , and therefore there are no observations available for node  $X_i$  at this time point. In addition, since we only have observations for  $X_i$  up to point  $m$ , the last observations of its parents cannot be used when computing the local score of  $X_i$ . We can therefore say that the effective sample size for computing a local score is  $m - 1$ .

## 2.2 Mixed Bayesian networks

Having established some fundamental ideas on already existing Bayesian network models, we can now introduce the concept of *mixed Bayesian networks*. Mixed Bayesian networks are probabilistic graphical models, which, similar to the previously discussed models, are described by two components: A network  $G$ , and a family of conditional probability distributions, with parameters  $q_i$ , associated with the nodes. The graph  $G(\mathcal{V}, \mathcal{E} \cup \tilde{\mathcal{E}})$  of a mixed Bayesian network (MBN) consists of

- $N$  static nodes in  $\mathcal{V}$  corresponding to the domain variables  $X_1, X_2, \dots, X_N$
- A set of directed edges  $\mathcal{E}$ . Every edge  $e_{ij} \in \mathcal{E}$  corresponds to a contemporaneous relation between nodes  $X_i$  and  $X_j$ . In other words, a correlation between  $X_i(t)$  and  $X_j(t)$  for every  $t \in T$  is implied.
- A set of directed edges  $\tilde{\mathcal{E}}$ . Every edge  $e_{ij} \in \tilde{\mathcal{E}}$  encodes a causal influence between node  $X_i$  and node  $X_j$ . Considering a time lag  $\tau$ , this implies that node  $X_i(t - \tau)$  affects the variable  $X_j(t)$  for every  $t \in T$ .

We will, from now on, refer to the edges in  $\mathcal{E}$  as the *static edges*, and to the edges in  $\tilde{\mathcal{E}}$  as the *dynamic edges* of the mixed Bayesian network. Similar to the dynamic Bayesian networks, the dynamic edges are allowed to form cycles- and self loops can apply as well. However, the spanning subgraph of  $G$  with edge set  $\mathcal{E}$  is required to be acyclic- that is, if we remove the dynamic edges in  $G$ , then the resulting network is a DAG. Therefore, if there is a cycle in  $G$ , then it must be formed by at least one edge in  $\tilde{\mathcal{E}}$ .

A more convenient representation of a mixed Bayesian network can be obtained if we add, for every node  $X \in \mathcal{V}$ , a dynamic counterpart  $\tilde{X}$ , which represents node  $X$ , but with a time lag. This induces a set  $\tilde{\mathcal{V}}$ , which contains the dynamic "clones" of the nodes, which represent previous configurations of the nodes. Therefore, in this representation of a mixed Bayesian network, a variable  $X_i$  corresponds to two nodes, one in  $\mathcal{V}$  and one in  $\tilde{\mathcal{V}}$ . Then, static edges can be identified as the edges with both endpoints in the same set, either  $\mathcal{V}$  or  $\tilde{\mathcal{V}}$ , whereas the dynamic nodes will have their starting point in  $\tilde{\mathcal{V}}$  and, point towards a node in  $\mathcal{V}$ .

If unfolded in time, a mixed BN can be visualized as a DAG with  $m$  layers, each layer corresponding to time point  $m$ . Then, the edges interconnecting layers (different time points) are the dynamic edges, while the edges within each layer are the static edges. There are no edges within the first layer, corresponding to  $t = 1$ . Given that every layer of the unfolded DAG is an exact copy of the last one, we will can represent an MBN compactly as a DAG with two layers, where the ordered

layers correspond to adjacent time points  $t-1$  and  $t$ .

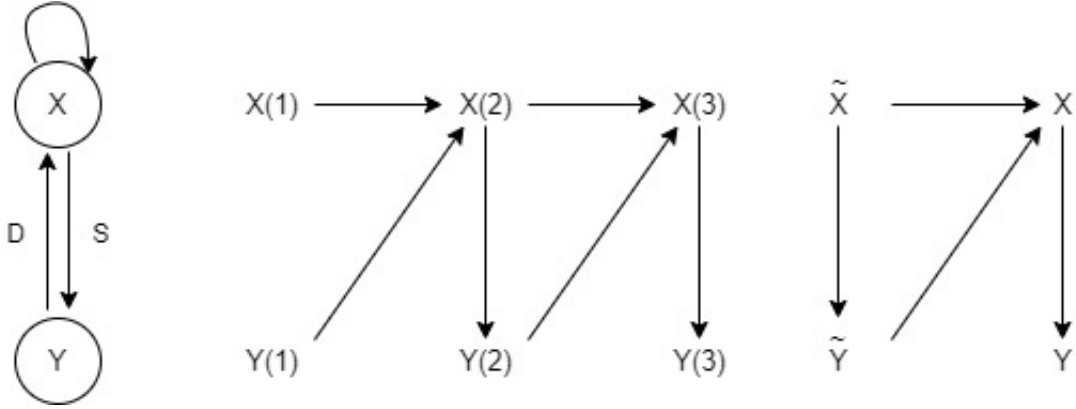


Figure 3: A *mixed Bayesian network* in three possible representations. The edge from  $X$  to  $Y$  is static and implies a contemporaneous relation. The edge from  $Y$  to  $X$  is a dynamic edge and implies an effect with a time lag  $\tau = 1$ . There is a self loop on node  $X$ .

We will now introduce an equivalent matrix representation of the adjacency relations of a mixed Bayesian network on  $N$  nodes. The Connectivity structure can be encoded into a  $2N \times N$  matrix  $A$ , consisting of two  $N \times N$  blocks. The upper block is the adjacency matrix of the induced sub graph  $G(\mathcal{V}, \mathcal{E})$ , while the lower block encodes the dynamic relations in the mixed network. Therefore, we have that if  $A(i, j) = 1$  and  $i \leq N$ , then there is a static edge from node  $i$  to node  $j$ . If  $A(i, j) = 1$  and  $i > N$ , then there is a dynamic edge from node  $i$  to  $j$ . We will refer to this matrix as the adjacency matrix, even though this term is not very accurate, since adjacency matrices are square matrices. Nevertheless in the mixed model a square matrix is not appropriate for representing the interactions between variables, since edges from  $\mathcal{V}$  to  $\tilde{\mathcal{V}}$  are invalid by construction, and edges between nodes in  $\tilde{\mathcal{V}}$  would correspond to contemporaneous relations, which are already encoded in the upper  $N \times N$  block. We can therefore interpret this matrix as follows: We extend the adjacency matrix  $G(\mathcal{V}, \mathcal{E})$  by adding  $N$ , time lagged clones of the nodes, which constitute candidate dynamic parents of the nodes  $X_1, X_2, \dots, X_N$ .

Mixed BNs with different topology can yield equivalent factorizations of the joint distribution. Thus, it is meaningful to define equivalence classes over the space of mixed BNs, in the sense that two MBNs are considered equivalent if and only if they imply the same conditional (in)dependencies (both contemporaneous and dynamic) between the domain variables.

For this cause, we define the mixed counterpart of a CPDAG, as it was described for the case of pure static observational data. The mixed-CPDAG of a mixed Bayesian network  $G(\mathcal{V}, \mathcal{E}, \tilde{\mathcal{E}})$  is a graph structure with the following properties:

- It has the same skeleton as  $G$ ,
- All dynamic edges are compelled, and are thus represented by directed edges in the CPDAG.
- All static edges that participate in a v-structure are compelled, and thus keep their direction in the CPDAG.

Although v-structures previously appeared only in static Bayesian network, in a mixed Bayesian network, a v-structure can be formed by both static and dynamic edges. More formally, v-structure in a mixed Bayesian network is either a subgraph of the form  $X_i(t) \rightarrow X_k(t) \leftarrow X_j(t)$  (two non adjacent nodes in  $\mathcal{V}$  converge to a third node in  $\mathcal{V}$ ), or of the form  $X_i(t-1) \rightarrow X_k(t) \leftarrow X_j(t)$  (two non adjacent nodes, one in  $\mathcal{V}$ , and one in  $\tilde{\mathcal{V}}$  converge to a third node in  $\mathcal{V}$ ).

It can be shown that two MBNs are in the same equivalence class, if and only if their mixed-CPDAGs coincide. An equivalence class in the space of mixed Bayesian networks can be uniquely represented by its mixed CPDAG.

It is not trivial to extract the CPDAG of a mixed Bayesian network. The challenge stems from the fact that we cannot directly apply Chickering's algorithm [10], like in a pure static network, because of the presence of dynamic edges. The dynamic edges are compelled by default, something that Chickering's algorithm cannot directly recognize, by construction. Consequently, edges that are compelled will possibly be misclassified as reversible, because they will be treated as if they were static. In the end of the section, we give a more thorough overview of the problem, and we explain how we can work through it, by introducing a trick we can apply before using Chickering's algorithm. With this trick we ensure that the algorithm will yield a correct classification of the static edges in a mixed Bayesian network.

In a mixed Bayesian network, every node is independent of its non-descendants, given its static parents  $\pi_i \in \mathcal{V}$  and its dynamic parents  $\tilde{\pi}_i \in \tilde{\mathcal{V}}$  with a time delay  $\tau = 1$ . Similar to the previously discussed models, the marginal likelihood can be decomposed into a product of local scores. The mixed counterpart of equations 6 and 8 is

$$P(D|G) = \prod_{i=1}^N \Psi[D_i^{\tilde{\pi}_i(t-1), \pi_i(t)}] \quad (9)$$

with  $D_i^{\tilde{\pi}_i(t-1), \pi_i(t)} = \{D_{i,t}, D_{\pi_i,t}, D_{\tilde{\pi}_i,t-1} : 2 \leq t \leq m\}$  contains those instances of data corresponding to the realizations of  $X_i$  from time point  $t = 2$  to time point  $t = m$ , the realizations of its static parents  $\pi_i$  over the same time points, and the realizations of its dynamic parents  $\tilde{\pi}_i(t-1)$ ,

from time point  $t = 1$  to time point  $t = m - 1$ . Similar to dynamic Bayesian networks, because of the time lag, the effective sample size for the computation of the local scores is  $m - 1$ . The local scores in the case of mixed Bayesian networks are given by:

$$\Psi[D_i^{\tilde{\pi}_i(t-1), \pi_i(t)}] = \int P(q_i | \pi_i, \tilde{\pi}_i) \prod_{t=2}^m P(X_i = D_{i,t} | \tilde{\pi}_i = D_{\tilde{\pi}_i, t-1}, \pi_i = D_{\pi_i, t}, q_i) dq_i \quad (10)$$

### Algorithm for Extracting the mixed CPDAG

To extract the CPDAG of a mixed BN we use a modified version of Chickering's DAG to CPDAG algorithm ([10]). This algorithm takes as input the incidence matrix of a mixed Bayesian network, and returns the incidence matrix of its CPDAG, where an undirected edge is interpreted as bidirectional. All dynamic edges are compelled, therefore, the lower block of the output matrix will be identical to the lower block of the input matrix. However, the algorithm will need to be aware of the dynamic relations, in order to detect those static edges that are rendered as compelled, because of the dynamic edges.

Consider the simple example illustrated in figure 4. In this example, the static edge from  $Z$  to  $Y$  is compelled: If reversed, a v-structure will be formed. Also, the dynamic edge is compelled by default. If we consider the depicted network as a static network, then we will not get the correct classification.

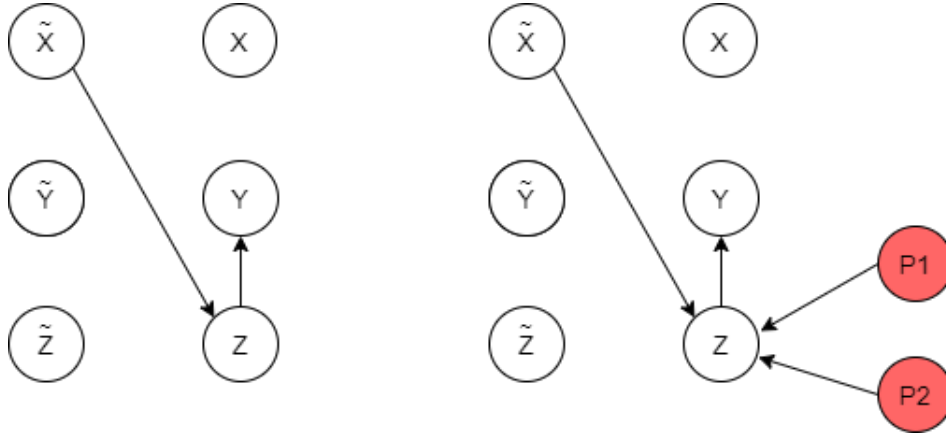


Figure 4: Addition of pseudoparents on the node with an incoming dynamic edge (right panel), in order to extract the CPDAG of the mixed BN on the left panel. The dynamic edge is compelled by default. The edge  $Z \rightarrow Y$  is compelled, because its reversal will yield a novel v-structure. Adding an artificial v-structure on  $Z$  ensures that both edges will be classified correctly

Although edge  $Z \rightarrow Y$  will be correctly classified as compelled, the dynamic edge will be seen as reversible from a "static point of view". If we, on the other hand, completely ignore the dynamic edges, for being compelled by default, then the remaining network will only consist of a single, static edge, which will again be wrongly classified as reversible.

To generalize, there are two cases that Chickering's algorithm does not cover for: Static edges, that participate in a v-structure with a dynamic edge, or static edges, whose reversal will result in the forming of a v-structure of one static and one dynamic edge. Both these cases can be averted, if we add a 'copy' of an incoming dynamic edge (a compelled edge) to every node with at least one dynamic parent.

To this end, we add two temporary pseudoparents  $P_1, P_2$  to the static part of the network. We then make every node  $X_i$  that has an incoming dynamic edge part of an artificial v-structure, by placing edges (pseudoedges) from the pseudoparents towards that node. One of the pseudoparents,  $P_1$ , is simply a copy of the dynamic parent. The second pseudoparent,  $P_2$  ensures that the edge  $P_1 \rightarrow X_i$  is compelled-it is, in other words, a proper copy of the incoming dynamic edge. In terms of matrices, we consider the static part of the adjacency matrix, and expand it by two rows and two columns corresponding to the pseudoparents. The result is an  $(N + 2) \times (N + 2)$  matrix, which we can plug in as an input to Chickering's algorithm. In our example, the expanded incidence matrix would be:

$$\begin{array}{c} X \\ Z \\ Y \\ P_1 \\ P_2 \end{array} \left( \begin{array}{ccc|cc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{array} \right)$$

The upper left block is the adjacency matrix of the static subgraph. The last two rows and the last two columns correspond to the artificial pseudoparents. Chickering's algorithm will return the  $(N + 2) \times (N + 2)$  matrix. We can then discard the part of this matrix that corresponds to the artificial parents, and only keep the upper left block. If we then staple it to the adjacency matrix of the dynamic subgraph, we will obtain the adjacency matrix of the mixed CPDAG. which, in our

example, will be equal to:

$$\begin{array}{c} X \\ Z \\ Y \\ \hline \tilde{X} \\ \tilde{Z} \\ \tilde{Y} \end{array} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

## 2.3 The Gaussian BGe scoring metric

Now that we have discussed the concept of Bayesian networks, we can directly address the problem of learning a Bayesian network from data. To this end, a scoring metric is required. The scoring metric can tell us how well a network matches the data, and it also allows us to compare between different networks, given the data. The scoring metric that we employ in this thesis is the BGe scoring metric of Geiger and Heckerman, which is thoroughly detailed in this subsection. Under some fairly weak assumptions, this metric allows us to derive a closed form solution for the marginal likelihood.

We will firstly discuss the standard linear Gaussian BGe scoring metric, as developed by Geiger and Heckerman ([19],[27]). Afterwards, we will describe how the BGe scoring metric has been adjusted into the dynamic setting under subsection 2.3.2. Finally, we will show how we can compute the BGe score of a mixed Bayesian network (subsection 2.3.3), after we perform some straightforward modifications.

### 2.3.1 Static Bayesian Networks

We consider the data to consist of  $m$  independent realizations of  $N$  variables, so we assume the data matrix  $D$  to be an  $N \times m$  matrix. The Gaussian BGe scoring metric assumes that the data comes from a multivariate normal distribution, with an unknown mean vector  $\mu$  and an unknown precision matrix  $W$ . The prior joint distribution on the parameters  $\mu, W$ , is assumed to be the normal Wishart distribution: The distribution of  $W$  is a Wishart distribution with  $\alpha > N - 1$  degrees of freedom and a positive definite precision matrix  $T_0$ . Also the conditional distribution of  $\mu$  given  $W$  is a normal distribution with mean  $\mu_0$  and a precision matrix  $vW$ , where  $v > 0$ . In theorem 3 of ([2]) it is proven that for this choice of prior distribution, the posterior joint distribution of  $\mu$  and  $W$  given the data is a multivariate normal distribution with a mean vector  $\tilde{\mu}$  and a precision matrix  $(v + m)W$ , with:

$$\tilde{\mu} = \frac{v\mu_0 + m\bar{D}}{v + m} \quad \bar{D} = \frac{1}{m} \sum_{j=1}^m D_{\cdot,j} \quad (11)$$

where  $D_{\cdot,j}$  is the  $j$ -th column of  $D$ . The marginal of  $W$  is then a Wishart distribution with  $a + m$  degrees of freedom and a precision matrix  $T_{D,m}$ , which is given by:

$$T_{D,m} = T_0 + \sum_{i=1}^m (D_{\cdot,i} - \bar{D})(D_{\cdot,i} - \bar{D})^T + \frac{vm}{v + m} (\mu_0 - \bar{D})(\mu_0 - \bar{D})^T \quad (12)$$

The hyperparameters  $T_0, \mu_0, \alpha, v$  have to be specified in advance. More specifically, any prior knowledge possessed by the user can be embodied into a *prior network*, which reflect an initial guess of the user on what the true underlying network might look like. This network can then be used to generate the hyperparameters  $T_0$  and  $\mu$ . The hyperparameters  $\alpha$  and  $v$  can be interpreted as the equivalent sample sizes for  $\mu$  and  $T_0$ , that is, the number of observations on which the initial guess of the user was based. For more details on the assessment of the hyperparameters, we refer the reader to [2], where a heuristic method is proposed by the authors. If the user's prior belief is that the unknown covariance matrix  $\Sigma = W^{-1}$  may be given by  $\Sigma_P$ , then  $T_0$  can be assessed as:

$$T_0 = \frac{v(a - N - 1)}{v + 1} \Sigma_P$$

If the user is ignorant about the unknown parameters  $W$  and  $\mu$ , then the effect of the prior distribution has to be minimized, which can be achieved by assigning suitable, uninformative values to the hyperparameters. The proposed setting is to take the covariance matrix  $\Sigma$  to be the identity matrix  $\mathbb{I}_N$ ,  $\mu_0$  to be an all-zero,  $N$ -dimensional vector,  $v$  and  $\alpha$  equal to 1 and  $N + 2$  respectively.

Under these assumptions, Geiger and Heckerman ([2]) derive the marginal likelihood of a *complete DAG*, that is a DAG in which every pair of nodes is joined with an edge. Thus, a complete DAG, denoted as  $G_C$  has the maximal number of edges, and every node is in direct interaction with any other. The equation of Geiger and Heckerman is given below:

$$P(D|G_C) = (2\pi)^{\frac{-Nm}{2}} \left( \frac{v}{v + m} \right)^{\frac{N}{2}} \frac{c(N, a)}{c(N, a + m)} \det(T_0)^{\frac{\alpha}{2}} \det(T_{D,m})^{-\frac{\alpha+m}{2}} \quad (13)$$

where  $\det(A)$  denotes is the determinant of matrix  $A$ . Moreover, and  $T_{D,m}$  is given in 12, and:

$$c(N, a) = \left( 2^{\frac{\alpha N}{2}} \pi^{\frac{N(N-1)}{4}} \prod_{i=1}^N \Gamma\left(\frac{\alpha + 1 - i}{2}\right) \right)^{-1}$$

where  $\Gamma(\cdot)$  is the Gamma function .

This methodology is now extended, so that the BGe score of an arbitrary graph can be computed. In theorem (2), Geiger and Heckerman [27], using the (fairly weak) assumptions of parameter independence and parameter modularity, it is proven that the marginal likelihood can be factorised

as follows:

$$\begin{aligned}
P(D|G) &= \prod_{i=1}^N \Psi(D_i^{\pi_i}) \\
&= \prod_{i=1}^N \frac{P(D^{\{X_i \cup \pi_i\}} | G_C(\{X_i \cup \pi_i\}))}{P(D^{\{\pi_i\}} | G_C(\{\pi_i\}))}
\end{aligned} \tag{14}$$

Where we as  $G_C(\{S\})$  we denote the complete graph on the nodes specified in the set  $S$  and  $\pi_i$  is the parent set of node  $i$ . Moreover,  $D^{\{S\}}$  stands for the submatrix of  $D$ , that is obtained from  $D$  by discarding all rows and all columns that correspond to nodes not in  $S$ . So, in order to compute the local score of node  $X_i$ , we need to restrict our attention on the sub domain, consisting only of  $X_i$  and the variables that correspond to the parents of  $X_i$ . Geiger and Heckerman ([19]) derive the term  $P(D^S | G_C(\{S\}))$ , where  $S$  is a subset of the variables in the domain:

$$P(D^{\{S\}} | G_C(\{S\})) = (2\pi)^{\frac{-|S|m}{2}} \left( \frac{v}{v+m} \right)^{\frac{|S|}{2}} \frac{c(|S|, a)}{c(|S|, a+m)} \det(T_0^S)^{\frac{\alpha}{2}} \det(T_{D,m}^S)^{-\frac{\alpha+m}{2}} \tag{15}$$

where  $T_{D,m}^S$  and  $T_0^S$  correspond to the sub matrices of  $T_{D,m}$  (given in 12) and  $T_0$ , which we obtain if we discard all the rows and all the columns corresponding in variables not in  $S$ .

This equation is, however, erroneous. As explained in [3], the correct equation is given by:

$$\begin{aligned}
P(D^{\{S\}} | G_C(\{S\})) &= (2\pi)^{\frac{-|S|m}{2}} \left( \frac{v}{v+m} \right)^{\frac{|S|}{2}} \frac{c(|S|, a - N + |S|)}{c(|S|, a + m - N + |S|)} \\
&\quad \times \det(T_0)^{\frac{\alpha - N + |S|}{2}} \det(T_m)^{-\frac{\alpha + m - N + |S|}{2}}
\end{aligned} \tag{16}$$

We note here that is case  $|S| = N$ , that is, if  $S$  contains all the variable in the domain, then equations 16 and 15 are identical. In other words, both equations give the same score for the complete graph on  $N$  nodes. The disagreement is detected when it comes to the scores of complete DAGs on a subset of nodes. The error stems from the distribution of the precision matrix  $W^s$ , which is the sub matrix of the precision matrix  $W$ , only focused on the variables contained in  $S$ . As detailed in the Supplementary material of [3], if  $W$  follows a Wishart distribution  $\mathcal{W}(\alpha, T_0)$ , then  $W^s$  follows a Wishart distribution, with covariance matrix  $T_0^S$  and  $a - N + |S|$  degrees of freedom. That is, the degrees of freedom of  $W_s$  are reduced according to the cardinality of  $S$ . For the derivation of 15, it is mistakenly assumed that  $W^s$  has the same degrees of freedom as  $W$ , namely,  $\alpha$  degrees of freedom.

### 2.3.2 Dynamic Bayesian Networks

We can straightforwardly modify the BGe scoring metric , to that is applicable in time series data[14]. Recall that in a Dynamic Bayesian network, an edge from  $X_i$  to  $X_j$  implies an interaction between  $X_i$  and  $X_j$ , but with a time delay  $\tau = 1$ .

The equivalent BGe score of a Dynamic Bayesian network can be derived if we follow the same computational steps as in the case of static Bayesian networks. In order to compute the local scores associated with the nodes, we will need to construct node-specific matrices,  $D(i)$ . To construct these matrices, we need to take into account the time lag, so that the realizations of the candidate parents of  $X_i$ , with a time delay  $X_j(t - 1)$  ,  $j \neq i$  are aligned with the realization of  $X_i(t)$  at the current time point  $t$ . The form of the matrix is different depending on whether a node is considered a candidate parent of itself (self loop). It is therefore meaningful to distinguish two cases: One where self loops are allowed, and one where self loops are denied.

#### Dynamic BGe score with self loops

In case self loops are considered valid edges, we will construct  $N + 1$  by  $m - 1$  node specific matrices, as illustrated below

$$D = \begin{bmatrix} D_{1,1} & D_{1,2} & \dots & D_{1,m} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1} & D_{N,2} & \dots & D_{N,m} \end{bmatrix} \quad \text{Original matrix } D$$

$$D(i) = \begin{bmatrix} D_{1,1} & D_{1,2} & \dots & D_{1,m-1} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1} & D_{N,2} & \dots & D_{N,m-1} \\ D_{i,2} & D_{i,3} & \dots & D_{i,m} \end{bmatrix} \quad \begin{array}{c} \text{Node specific matrix in case} \\ \text{of self loops} \end{array}$$

Note that the first  $N$  rows correspond to the potential parents of  $X_i$ , which are subject to a time delay. Because of the time delay, we only focus on the first  $m - 1$  measurements for these variables. They can be interpreted as predictors for  $X_i(t + 1)$  . Since self-loops are considered valid,  $X_i(t)$  is also a candidate parent. Note that an extra, time shifted row is added to the matrix, which corresponds to  $X_i(t + 1)$ . We can identify this additional row as the dependent variable. Again, because of the time lag, no information is available for  $X_i$  at time point  $t = 1$ , thus , its first realization is discarded.

Using the same reasoning as in the static case , we find that the marginal likelihood in this model,

is given by:

$$\begin{aligned}
P(D|G) &= \prod_{i=1}^N \Psi[D_i^{\pi_i(t-1)}] \\
&= \prod_{i=1}^N \frac{P(D(i)^{\{X_{N+1}, \pi_i\}} | G_C(\{X_{N+1}, \pi_i\}))}{P(D(i)^{\{\pi_i\}} | G_C(\{\pi_i\}))}
\end{aligned} \tag{17}$$

where  $G_C(\{S\})$  denotes the complete graph on the nodes specified in  $S$ , and  $\pi_i$  corresponds to the parents of  $X_i$ . The variable of  $X_{N+1}$  corresponds to the extra variable that we add to the node specific data matrix, while  $\Psi[D_i^{\pi_i(t-1)}]$  is given by 7.

### Dynamic BGe score without self loops

In case self loops are not considered valid edges, we need to construct  $N$  by  $(m-1)$  node specific matrices  $D(i)$ , following the same logic as above.

$$\begin{aligned}
D &= \begin{bmatrix} D_{1,1} & D_{1,2} & \dots & D_{1,m} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1} & D_{N,2} & \dots & D_{N,m} \end{bmatrix} & D(i) = \begin{bmatrix} D_{1,1} & D_{1,2} & \dots & D_{1,m-1} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ D_{i-1,1} & D_{i-1,2} & \dots & D_{i-1,m-1} \\ D_{i,2} & D_{i,3} & \dots & D_{i,m} \\ D_{i+1,1} & D_{i+1,2} & \dots & D_{i+1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1} & D_{N,2} & \dots & D_{N,m-1} \end{bmatrix} \\
&\text{Original matrix } D & \text{Node specific matrix } D(i) \text{ in case} \\
& & \text{of no self-loops}
\end{aligned}$$

In this case, the  $i$ -th row is time shifted, and it corresponds to the dependent variable  $X_i(t+1)$ . The first  $i-1$  and the last  $N-i$  rows of  $D(i)$  correspond to the predictor variables  $X_j(t)$ , which are subject to a time lag.

Following the same computational steps, we find that the marginal likelihood in this model, is

given by:

$$\begin{aligned}
P(D|G) &= \prod_{i=1}^N \Psi[D_i^{\pi_i(t-1)}] \\
&= \prod_{i=1}^N \frac{P(D(i)^{\{X_i, \pi_i\}} | G_C(\{X_i, \pi_i\}))}{P(D(i)^{\{\pi_i\}} | G_C(\{\pi_i\}))}
\end{aligned} \tag{18}$$

where  $G_C(\{S\})$  denotes the complete graph on the nodes specified in  $S$ , and  $\pi_i$  corresponds to the parents of  $X_i$ . Moreover,  $\Psi[D_i^{\pi_i(t-1)}]$  is given by 7.

The score of the complete graph for dynamic Bayesian networks is given by:

$$\begin{aligned}
P(D(i)^{\{S\}} | G_C(\{S\})) &= (2\pi)^{\frac{-|S|(m-1)}{2}} \left( \frac{v}{v + (m-1)} \right)^{\frac{|S|}{2}} \frac{c(|S|, a - \tilde{N} + |S|)}{c(|S|, a + (m-1) - \tilde{N} + |S|)} \\
&\times \det(T_0^{\{S\}})^{\frac{\alpha - \tilde{N} + |S|}{2}} \det(T_{D(i), m-1}^{\{S\}})^{-\frac{\alpha + (m-1) - \tilde{N} + |S|}{2}}
\end{aligned} \tag{19}$$

Where  $\tilde{N}$  is equal to the first dimension of the node specific matrices- that is  $\tilde{N} = N + 1$  if self loops are allowed and  $\tilde{N} = N$  otherwise.  $|S|$  corresponds to the number of variables in the subset  $S$ , and  $T_0^{\{S\}}$ ,  $T_{D(i), m-1}^{\{S\}}$  are the submatrices of  $T_0$  and  $T_{D(i), m-1}$  respectively, that only consist of those rows and those columns that correspond to the variables in  $S$ . The node specific covariance matrix  $T_{D(i), m-1}$  is given as:

$$T_{D(i), m-1} = T_0 + \sum_{i=1}^{m-1} (D(i)_{.,j} - \bar{D}(i))(D(i)_{.,j} - \bar{D}(i))^T + \frac{v(m-1)}{v + (m-1)} (\mu_0 - \bar{D}(i))(\mu_0 - \bar{D}(i))^T \tag{20}$$

where if we denote the  $j$ -th column of  $D$  by  $D_{.,j}$ , we symbolize:

$$\bar{D}(i) = \frac{1}{m-1} \sum_{j=1}^{m-1} D_{.,j}$$

### 2.3.3 Mixed Bayesian networks

We are interested in calculating the BGe score of a specific mixed Gaussian network  $G$ , given a collection of mixture data  $D = (X_1(t), X_2(t), \dots, X_N(t))_{t=1}^m$ . Similar to the dynamic case, we will define a modified node-specific matrix  $D(i)$ , which will allow us to follow the same computational steps of the BGe score, as in the static case.

Again, taking the time lag into account, we will only focus on time points  $t = \tau + 1, \dots, m$ . Assuming a time lag of  $\tau = 1$ , we have that the realization of  $X_i(t)$  is influenced not only by the realizations of its dynamic parents at time point  $t-1$ , but also by the realizations of its static parents at the current time point  $t$ . Therefore, we will extend the node specific matrix, so that we include all potential parents of node  $X_i(t)$ . This will result in a two-block matrix: The upper  $N \times m - 1$  block will represent the "static part" of the data. The lower block, whose first dimension will vary depending on whether we allow or deny self loops, will represent the "dynamic part" of the data. There, the time-delayed (shifted) realizations of the nodes can be found. After constructing the two parts (in a manner which we will specify in a moment), the mixed data matrix will be obtained by "stapling" them-that is, by appending the static part with the dynamic part.

Regardless of the absence or presence of self loops, we can obtain the static part of the data, simply by discarding the first column, corresponding to the realizations of the nodes at time point  $t = 1$ . For the dynamic part, we will follow a similar logic as in the pure dynamic case. Hence, the same two cases need to be distinguished:

#### BGe score of mixed Bayesian networks without self loops

If self-loops are denied, the dynamic part of the mixed data will be an  $N \times m - 1$  matrix. However after stapling, row  $i$  is a duplicate, as it is identical to row  $N+i$ . Hence, row  $N+i$  is discarded, which results in a  $(2N - 1) \times (m - 1)$  node specific matrix.

$$D = \begin{bmatrix} D_{1,1} & D_{1,2} & \dots & D_{1,m} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1} & D_{N,2} & \dots & D_{N,m} \end{bmatrix}$$

Original matrix  $D$

$$D(i) = \begin{bmatrix} D_{1,2} & D_{1,3} & \dots & D_{1,m} \\ D_{2,2} & D_{2,3} & \dots & D_{2,m} \\ \vdots & \dots & \ddots & \vdots \\ D_{N,2} & D_{N,3} & \dots & D_{N,m} \\ D_{1,1} & D_{1,2} & \dots & D_{1,m-1} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ D_{i-1,1} & D_{i-1,2} & \dots & D_{i-1,m-1} \\ D_{i+1,1} & D_{i+1,2} & \dots & D_{i+1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1} & D_{N,2} & \dots & D_{N,m-1} \end{bmatrix}$$

Node specific matrix in case  
of no self loops

The dependent variable  $X_i(t+1)$  lies in row  $i$  of the matrix, while the rest of the rows corresponds to candidate explanatory variables. The variables in the static part,  $X_j, j \neq i, j \leq N$  represent the contemporaneous predictors  $X_j(t+1)$ , while the variables corresponding to the last  $N-1$  rows of the matrix  $X_j, j > N$  correspond the candidate dynamic predictors  $X_j(t)$ .

### BGe score of mixed Bayesian networks with self loops

When self-loops are allowed, the dynamic part of the mixed data will be an  $(N+1) \times (m-1)$  matrix. After stapling the two matrices, row  $i$  is duplicated and identical to row  $2N+1$ . Hence, the last row of the matrix is discarded, which results in a  $(2N) \times (m-1)$  node specific matrix.

$$D = \begin{bmatrix} D_{1,1} & D_{1,2} & \dots & D_{1,m} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1} & D_{N,2} & \dots & D_{N,m} \end{bmatrix}$$

Original matrix  $D$

$$D(i) = \begin{bmatrix} D_{1,2} & D_{1,3} & \dots & D_{1,m} \\ D_{2,2} & D_{2,3} & \dots & D_{2,m} \\ \vdots & \dots & \ddots & \vdots \\ D_{N,2} & D_{N,3} & \dots & D_{N,m} \\ D_{1,1} & D_{1,2} & \dots & D_{1,m-1} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N,1} & D_{N,2} & \dots & D_{N,m-1} \end{bmatrix}$$

Node specific matrix in case  
of self loops

Note here that, if self loops are considered valid edges, then the node-specific matrix then we have  $D(i) = D(j)$  for all nodes  $X_i, X_j$ .

In order to derive the Gaussian score of a mixed Bayesian network, given the mixed data matrix  $D$ , we want to evaluate the terms in the product (local scores):

$$P(D|G) = \prod_{i=1}^N \frac{P(D(i)^{\{X_i, \pi_i, \tilde{\pi}_i\}} | G_C(\{X_i, \pi_i, \tilde{\pi}_i\}))}{P(D(i)^{\{\pi_i, \tilde{\pi}_i\}} | G_C(\{\pi_i, \tilde{\pi}_i\}))} \quad (21)$$

where we denote as  $\pi_i$ , We define  $D(i)^S$ , is the sub-matrix of the matrix  $D(i)$ , consisting only of the rows corresponding to the nodes included in  $S$ . Under our assumptions, the BGe score for mixed Gaussian networks can be computed similarly to the static BGe score. Therefore the score of the complete graph, regardless the presence of self loops is given by

$$\begin{aligned} P(D(i)^{\{S\}} | G_C(\{S\})) &= (2\pi)^{\frac{-|S|(m-1)}{2}} \left( \frac{v}{v + (m-1)} \right)^{\frac{|S|}{2}} \frac{c(|S|, a - \tilde{N} + |S|)}{c(|S|, a + (m-1) - \tilde{N} + |S|)} \quad (22) \\ &\times \det(T_0^{\{S\}})^{\frac{\alpha - \tilde{N} + |S|}{2}} \det(T_{D(i), m-1}^{\{S\}})^{-\frac{\alpha + (m-1) - \tilde{N} + |S|}{2}} \end{aligned}$$

In the case where self loops are not regarded as valid edges, we have and  $\tilde{N} = 2N - 1$ . If self loops are allowed, and  $\tilde{N} = 2N$ .

Here,  $T_{D(i), m-1}^S, T_0^S$  are the square sub-matrices of the square matrices  $T_{D(i), m-1}, T_0$  respectively, which consist only of the rows and columns in  $S$ . The matrix  $T_{D(i), m-1}$  is given as

$$T_{D(i), m-1} = T_0 + \sum_{i=1}^{m-1} (D(i)_{.,j} - \bar{D}(i))(D(i)_{.,j} - \bar{D}(i))^T + \frac{v(m-1)}{v + (m-1)} (\mu_0 - \bar{D}(i))(\mu_0 - \bar{D}(i))^T$$

## 2.4 Markov Chain Monte Carlo

### 2.4.1 Motivation

Our objective is, to determine the posterior probability of a feature  $f$  over all possible network structures  $G$ . This is equal to:

$$P(f|D) = \sum_G P(G|D)I(G, f) \quad (23)$$

where

$$I(G, f) = \begin{cases} 1 & \text{if } f \text{ exists in } G \\ 0 & \text{otherwise} \end{cases}$$

Nevertheless, the exhaustive enumeration of all network structures is not viable, even in low dimensions, because the number of networks increases super exponentially on the number of nodes  $N$ , which creates a computational overhead.

A reasonable strategy is to approximate this sum by reducing the space of network structures on a subset of high scoring networks. A simple and obvious approach, is to take this subset to consist of the single network structure that maximizes the likelihood  $P(D|G)$ . Greedy search heuristics algorithms can be employed for this task. However, this approach is not always optimal, since they assume that the posterior is peaked around one single model, which is seldom the case. In most cases there are several structures with non-negligible posteriors, that is, there exist multiple local optima that explain the data sufficiently well. Hence, determining a single, top-scoring structure might prove unreliable when it comes to determining the probability of a feature in the model. Therefore, a preferable strategy is to average over a larger set of structures with high posterior probabilities.

In this section, we will present two alternative approaches that will allow us to obtain such a sample of networks, in subsection 2.4.4. One is *Structure MCMC* of Madigan and York [28], and the second is *Order MCMC* of Friedman and Koller [13]. In the following subsection 2.4.2, we recapitulate some mathematical theory behind *Markov Chain Monte Carlo* and the *Metropolis-Hastings* algorithm, which is essential for the MCMC techniques we will later employ.

## 2.4.2 Mathematical Background

In order to understand the concept of Markov chain Monte Carlo and the Metropolis-Hastings algorithm, we need to establish some essential preliminary concepts on discrete Markov chains.

**Definition 2.1.** A discrete **Markov Chain** of first order is a stochastic process  $\{X_t\}_{t=1,2,\dots}$  with the discrete state-space  $S$ , that has the property

$$P(X_t = x_t | X_{t-1} = x_{t-1}, \dots, X_1 = x_1) = P(X_t = x_t | X_{t-1} = x_{t-1})$$

for every state  $x_t \in S$

For a discrete Markov chain we can, without loss of generality, write  $S = \{1, 2, \dots, k\}$ , where  $k = |S|$ . The definition tells us that the state  $x_{t+1}$  depends only the previous state  $x_t$ . We will refer to this property as the *local Markov assumption*. A Markov chain is fully characterized by a *transition kernel*  $T(x, y) := P(X_t = y | X_{t-1} = x)$  and an initial, prior distribution  $p_0$ . We can thus simulate a Markov chain, by choosing an initial state, sampled from the initial distribution  $p_0$ , and then sample every new state  $x_{t+1}$  according to the distribution  $P(x_{t+1} | X_t = x_t)$ , which is specified by the transition kernel. Such a collection of states is called a *trajectory*.

Therefore, the transition kernel defines a distribution over the succeeding states. This distribution is usually sensitive to the initial state. However, under certain conditions, and only after a sufficient number of time steps the chain converges to a *stationary distribution*  $\pi$ . This distribution is *time invariant*, meaning that after convergence, all succeeding states are drawn according to the stationary distribution  $\pi$ .

A stationary distribution does not always exist, and if it does, it is not always unique. Convergence requires that a Markov chain is *ergodic*. The ergodicity of a chain is implied by two other properties: **Aperiodicity** and **irreducibility**:

**Definition 2.2.** A state in  $S$  is called *periodic* with period  $k$  if any return to this state occurs in multiple times  $k$  steps. A state that does not have this property is called *aperiodic*. A Markov chain is called **aperiodic** if all of its states are aperiodic.

This definition a discrete Markov chain with finite state-space  $S$  is aperiodic if the probability of reaching any state, in an arbitrary, though sufficiently large number of steps, is always non zero.

**Definition 2.3.** A discrete Markov chain with finite state-space is called **irreducible** if there is a

$t > 0$  such that  $P(X_t = j | X_1 = i) > 0$ , for every  $i, j \in S$

More intuitively, a discrete Markov chain with finite state-space is called **irreducible**, if every state in its state-space is reachable from any other state by a sequence of transitions that have positive probability, that is, if every two states *communicate* with each other.

The ergodicity of a Markov chain guarantees that, if a stationary distribution exists, then it is unique. However, there is an additional requirement to convergence, and that is *reversibility*.

**Definition 2.4.** A discrete Markov chain with finite state-space is called **reversible** if its stationary distribution satisfies the **Equation of detailed Balance**

$$\frac{T_{i,j}}{T_{j,i}} = \frac{p(j)}{p(i)}, \quad \forall i, j \in S$$

It is also easy to prove that Detailed balance implies invariance, that is, if a distribution  $\pi = (p(1), p(2), \dots, p(k))$  fulfils the Equation of Detailed Balance, then the distribution  $\pi$  is the stationary distribution of this ergodic chain. Under ergodicity, a reversible Markov chain inevitably converges to its stationary distribution.

### 2.4.3 The MCMC scheme and the Metropolis-Hastings algorithm

In a nutshell, the idea of MCMC is to find a transition kernel of an ergodic Markov chain that has the posterior distribution as its (unique) stationary distribution. Then, hopefully, if the constructed chain is simulated for long enough, we will be able to draw samples from a sensibly chosen distribution, so that they approximate samples from the target distribution. To do so, we will employ the *Metropolis Hastings* algorithm as described by Hastings in [30].

Initially, a prior distribution  $p(\cdot)$  over the state-space needs to be specified. The algorithm starts from an initial state  $x_0$ . In every step, a new succeeding state  $i$  is proposed with probability  $Q(i|x_t)$ . The newly proposed state is accepted with probability:

$$A(i|j) = \min \left\{ \frac{p(j) Q(j|i)}{p(i) Q(i|j)}, 1 \right\}$$

If the new state is rejected, then the chain remains to its current state. Below we show the algorithm in pseudocode:

---

**Initialize:**  $x_0$ , randomly selected from  $S$

**for**  $t = 1, 2, \dots$  **do**

    Given  $x_t = j \in S$ , propose succeeding state  $x_{t+1} = i \in S$  with probability  $Q(i, j)$

    Draw a random number  $c \in [0, 1]$

**if**  $c < A(i|j)$  **then**

        | accept new state and set  $x_{t+1} = i$

**else**

        | reject new state and set  $x_{t+1} = j$

**end if**

**end for**

---

Note that the transition probability from state  $i$  to state  $j$  is equal to the probability of first proposing and then accepting  $j$ . The probability of staying to  $i$  is then the probability that the new state  $j \neq i$  is proposed and then rejected. Formally we have:

$$T(i, j) = \begin{cases} A(i, j) Q(i, j) & i \neq j \\ 1 - \sum_{j \neq i} A(i, j) Q(i, j) & \text{otherwise} \end{cases}$$

If we make a sensible choice of the proposal probabilities  $Q(\cdot, \cdot)$ , that is, a choice that conserves the ergodicity of the chain, then it can easily be seen that the transition kernel satisfies the Equation of detailed Balance, which also establishes reversibility. Both these conditions ensure that the chain will eventually converge to its stationary distribution.

When sampling from a Markov chain, we need to take into account the *autocorrelation* of the chain. Theoretically, from the Markov property, every sample is only dependent on its preceding state. This, however, will possibly induce a positive correlation between a sample and other samples within a window  $d$ . In order to avert this problem, a reasonable strategy is to subsample on the chain, that is, to sample every  $n$  steps of the simulation, for a positive integer  $n$ . This is termed as *thinning* of the Markov chain. Another commonly used strategy is to discard the states sampled in the early phase of the simulation, and start gathering samples only after the chain is hopefully closer to convergence. We call this early phase the *burn-in* phase. This approach will allow us to obtain samples that are close to independent.

#### 2.4.4 MCMC sampling of Bayesian networks

Now that the foundation of MCMC and the Metropolis Hastings algorithm has been presented, we can discuss how we can adopt these techniques for our goal, which is, to learn a network structure from a domain. We present two different schemes, *structure-MCMC* and *order-MCMC*. Although Structure-MCMC (2.4.4.1) performs very well in smaller domains, it comes with convergence issues in larger domains, that is, as the number of variables or the number of observations increase. In these scenarios, we can employ the alternative approach, *order-MCMC* (2.4.4.2), which can improve convergence, with the price of introducing a small amount of bias.

##### 2.4.4.1 Structure MCMC

Structure MCMC is a sampling scheme proposed by Madigan and York, for sampling structures from the posterior distribution, using a Metropolis-Hastings sampler. To that end, a Markov chain is defined over the space of structures, that has the posterior distribution  $P(G|D)$  as its stationary distribution. This Markov chain can then serve as a generative mechanism of samples  $G_1, G_2, \dots, G_T$  from the posterior. Provided that we have simulated the chain long enough for the stationary distribution to be reached, we will hopefully obtain a collection of structures that are representative of the posterior.

In the context of static Bayesian networks, we consider the state space  $S$  to be the space of DAGs. We will afterwards describe the modified scheme that applied to Dynamic and Mixed Bayesian networks. The general structure MCMC process is summarized below in pseudocode 1:

---

**Algorithm 1** Structure-MCMC sampling scheme

---

**Initialize:**  $G_1$ , selected from  $S$

**for**  $t = 1, 2, \dots$  **do**

    Given  $G_t$ , propose new state  $G \in S$  with probability  $Q(G, G_t)$

    Accept network  $G$  with probability  $A(G, G_t)$

**if**  $G$  is accepted **then**

        | set  $G_{t+1} = G$

**else**

        | set  $G_{t+1} = G_t$

**end if**

**end for**

---

In every step of the MCMC simulation, a succeeding state is proposed. An operator that determines how succeeding states are proposed needs to be specified- for structure MCMC, we use

*valid edge operations* to transit to new states. In the context of static Bayesian networks, valid edge operations are edge *additions*, *deletions* or *reversals* that lead to a valid (acyclic) DAG. We define the *neighbourhood* of  $G$ , denoted  $N(G)$ , to be the family of DAGs that can be reached from  $G$ , by a valid edge operation.

In order to design a Markov chain in the space of DAGs, we need to specify the transition kernel of this chain, denoted as  $T(G, G^*)$ , in a way that guarantees that the resulting chain has the desired properties. In other words, we need to ensure that the transition kernel satisfies the equation of Detailed Balance.

For the transition kernel, we have:

$$T(G^*, G) = \begin{cases} Q(G^*, G) A(G^*, G), & \text{if } G^* \in N(G) \\ 1 - \sum_{G' \in N(G)} Q(G', G) A(G', G) & \text{if } G^* = G \end{cases}$$

Therefore, the problem reduces to choosing an appropriate mechanism of proposing and accepting new states. For the proposal probability, a common strategy is to take all reachable states (*neighbor graphs*) from  $G$  to be equally likely to be proposed. Therefore, if we denote  $N(G)$  the set of all neighbours of  $G$ , then we define the proposal probability as:

$$Q(G, G^*) = \begin{cases} \frac{1}{|N(G)|} & \text{if } G^* \in N(G) \\ 0 & \text{otherwise} \end{cases}$$

To ensure the ergodicity and reversibility of our chain, it suffices to define the acceptance probability  $A(G, G^*)$  so that the transition kernel from  $G$  to  $G^* \in N(G)$ , satisfies the equation of Detailed Balance.

It can easily be shown that taking:

$$\begin{aligned} A(G, G^*) &= \min \left\{ 1, \frac{P(G^*|D) Q(G^*, G)}{P(G|D) Q(G, G^*)} \right\} \\ &= \min \left\{ 1, \frac{P(G^*|D)P(G^*)}{P(G|D)P(G)} \frac{|N(G)|}{|N(G^*)|} \right\} \end{aligned} \tag{24}$$

will result in a chain that has the desired stationary distribution.

Structure MCMC can be adjusted for the dynamic and the mixed setting- what changes is the

state space, and the rule (operator) that defines the neighbourhood  $N(G)$  of a structure  $G$ .

In the context of dynamic Bayesian networks, the state space includes all directed networks, with or without self loops, depending on whether we consider self loops valid or invalid edges. The neighbourhood of a directed network,  $N(G)$ , is the set of networks that we can obtain from  $G$  by one single edge *addition* or *deletion*. In contrast to static BNs, the states are not subject to the acyclicity limitation, therefore all edge additions that result in a directed network in the state-space are possible. In static and mixed Bayesian networks we need to be more cautious when it comes to either adding or reversing an edge, as these moves might result in the forming of an invalid cycle. Edge deletions on the other hand, are always valid in both models.

An additional constraint that is commonly adopted in structure MCMC is what we call the *fan-in* restriction. This restriction imposes an upper bound to the number of parents a node can have. Thus, a network complies with a fan in constraint  $k$ , if all nodes in the network have at most  $k$  incoming edges. In every structure MCMC step, all neighbour graphs that violate the fan-in restriction, are excluded from the neighbourhood, and are thus not considered as candidate succeeding states in the chain. Imposing a fan-in restriction shrinks the search space, as the number of possible networks drops significantly. As a consequence, the convergence rate of the Markov chain is improved. Moreover, the fan in restriction comes with reduced computational costs.

The standard MCMC scheme can straightforwardly be adjusted in the space of mixed Bayesian networks as well. The search space is the space of mixed Bayesian networks, and the neighbourhood of a mixed BN  $G$  is defined as family of mixed BNs that can be reached from  $G$  with an edge addition, and edge removal and a reversal of a static edge. Keeping in mind the restrictions imposed in the two previous models, we summarize:

- An edge addition is valid if it is valid in the induced static subgraph of the mixed network, and if it does not violate the self loops and fan-in restriction.
- An edge removal is always valid.
- An edge reversal is valid if it is valid in the induced static subgraph of the mixed network, and if it does not violate the fan-in restriction.

The determination of the number of iterations is crucial. The optimal number of iterations depends strongly on the size of the network, since a larger number of nodes implies a larger state-space. It also depends on the shape of the posterior distribution. The smoother the posterior, the more likely it is for the constructed chain to reach faster convergence and mixing. When applying structure-

MCMC, we need to ensure that we adjust the number of iterations properly, so that, if independent simulations are run on the same data set, then similar estimates of the posterior of the edges are obtained. Convergence can be monitored using scatter plots of the edge scores obtained by two independent simulations. If sufficient convergence has been achieved, then the points in the plot will be close to the diagonal.

While structure MCMC performs well in relatively small domains, it is rather slow in mixing and convergence as dimensions increase. The size of the search space, as well as the nature of the posterior, which, in realistic scenarios, is not so smooth and suffers from multiple local optima, hinders convergence. For these reasons, significant convergence problems are expected in case structure MCMC is employed for the inference of a mixed Bayesian network: In a mixed Bayesian network, the nodes are doubled, because the possible, dynamic parents are added. To this end, we turn to an alternative MCMC approach, called *order-MCMC*, which was proposed by Friedman and Koller 2.4.4.2. For reasons that will shortly be discussed, order MCMC leads to a faster and more reliable convergence to the target distribution.

#### 2.4.4.2 Order MCMC

*Order MCMC*, ([13]) is a variation of the MCMC algorithm, only, instead of constructing the Markov chain in the space of structures, order MCMC constructs a Markov chain in the space of *topological orders*. An order ( $\prec$ ) is a total ordering of the nodes, such that if  $X_i \prec X_j$ , then if there is an edge between those two variables, it must be directed from  $X_i$  to  $X_j$ . In other words, an order is a permutation  $\sigma$  of the nodes, such that if  $X_i$  is an ancestor of  $X_j$ , then  $\sigma(i) < \sigma(j)$ . Denoting  $X_{\sigma(j)} = i_j$ , for all  $j \in [n]$ , we will, for simplicity, write an order as  $(i_1, i_2, \dots, i_n)$ . We say that a parent set for node  $j$  is consistent with an order, if all nodes forming this parent set stand in the left of  $j$  in the order. Similarly, a network is consistent with an order, if the parent set of every node in the network is consistent with the order. With this in mind, we can see an order as a compact representation of the networks that are consistent with it.

The goal of order MCMC is to generate a sample of orders from the posterior distribution  $P(\prec | D)$  over the space of orders. Assuming that a fan-in restriction  $k$  has been imposed, then the likelihood

of an order given the data is given by

$$\begin{aligned}
P(D | \prec) &= \sum_{G \in \mathcal{G}_{k, \prec}} P(G | \prec) P(D | G) \\
&= \sum_{G \in \mathcal{G}_{k, \prec}} \prod_i \text{score}(X_i, \text{Pa}_G(X_i) | D) \\
&= \prod_i \sum_{U \in \mathcal{U}_{i, \prec}} \text{score}(X_i, U | D).
\end{aligned} \tag{25}$$

We denote  $\text{score}(X_i, U | D)$  = the local score of  $X_i$ , which depends on the model we employ. In addition,  $\mathcal{G}_k$  represents the family of networks that are consistent with  $\prec$  and subject to a fan-in restriction of  $k$ , and

$$\mathcal{U}_{i, \prec} = \{U : U \prec i, |U| \leq k\}$$

Equation 25 tells us that a fan-in restriction  $k$  is essential in order MCMC. As the sum goes over all possible parent sets that are consistent with the order, we need the fan in restriction so that the number of consistent parent sets does not explode.

In every step of the order-MCMC, a new order is proposed, and then either accepted or rejected, according to the standard Metropolis algorithm. The recommended operation for proposing a new order is node flipping. That is, two nodes are randomly chosen and they exchange places in the order, while the others remain unchanged:

$$(i_1, \dots, i_j, \dots, i_k, \dots, i_n) \rightarrow (i_1, \dots, i_k, \dots, i_j, \dots, i_n)$$

For the construction of the Markov chain, the proposal probabilities  $Q(\prec' | \prec)$  need to be specified, so that the ergodicity and reversibility of the chain are guaranteed. We then accept the newly proposed order with probability equal to:

$$A(\prec', \prec) = \min \left\{ 1, \frac{P(\prec' | D) Q(\prec | \prec')}{P(\prec | D) Q(\prec' | \prec)} \right\}. \tag{26}$$

This results in a Markov chain that will converge to the target distribution. The authors have assumed a uniform distribution over the space of orders, so we have,  $P(\prec) = \frac{1}{n!}$ . The probability of sampling  $\prec'$  from the reachable states of  $\prec$ , that is, the orders that can be obtained from  $\prec$  with one flipping operation is equal to  $\frac{2}{n(n-1)}$ . Consequently, in contrast to structure-MCMC, in order MCMC we have symmetric proposal probabilities. The terms cancel out, which reduces

the acceptance probability of a newly proposed order to:

$$A(\prec', \prec) = \min \left\{ 1, \frac{P(\prec' | D)}{P(\prec | D)} \right\}. \quad (27)$$

Apart from the fan in restriction  $k$ , Friedman and Koller [13] propose various computational bottlenecks to improve the efficiency of the model. In the recommended approximation, not all possible parent sets are utilized. A number  $C$  of highest scoring parents from every node is specified, and then the set of possible parent sets for every node is reduced to all possible combinations of those parents. Given a fan-in constraint on the cardinality of the parent sets, this will result in  $\sum_{s=1}^k \binom{C}{s}$  possible parent sets for every node, whose scores are computed and stored before the MCMC simulation commences. This bottleneck reduces running time significantly, especially in larger domains. It does have some shortcomings though, as high scoring parent sets can be left out. In case there is not a significant difference between the highest and the lowest parent set for a node, the authors propose what they refer to as a full enumeration of the parent sets.

Another practical shortcut is recommended for the computation of the score of a newly proposed order. Given that every new order results from flipping two nodes in the order, then the parent sets that are affected are the ones of the nodes that stand in between, while there is no impact on the nodes standing before and after the flipped nodes. Therefore, we just need to detect the valid parent sets of only those nodes in order to compute the score of the new order.

Given a sample of orders  $\prec_1, \prec_2, \dots, \prec_T$  from the posterior  $P(\prec | D)$ , a sample of networks can be obtained following a simple sampling approach ([13]). For every node, a parent set can be sampled from the distribution

$$P(Pa_G(X_i) | \prec, D) = \frac{score(X_i, U | D)}{\sum_{U' \in \mathcal{U}_i} score(X_i, U' | D)} \quad (28)$$

which will generate an entire network, that is consistent with the order. Higher scoring families and, consequently, higher scoring networks are more likely to be sampled. This is a well defined sampling scheme, in the sense that it will always result in a valid network in the space of DAGs. When we sample a valid parent set for  $X_i$ , which contains  $X_j$ , this implies by default that  $X_j$  precedes  $X_i$  in the order. It is therefore impossible for the sampled parent sets to form a cycle, since all possible parent sets of  $X_j$  stand on the left of  $X_j$ , and thus, of  $X_i$  in the order. Therefore, we can sample a family for every node independently from the others. This important observation is

also implied in 25: Summing over all networks that are consisted with an order  $\prec$  is equivalent to summing over all possible families for every node and then multiplying over the nodes.

Order-MCMC can easily be adjusted for the the new mixed model. We recall that a mixed Bayesian network consists of the original set of nodes,  $\mathcal{V}$ , and a duplicate set of potential dynamic parents  $\tilde{\mathcal{V}}$ . An order in the modified model is an ordering of the nodes contained in  $\mathcal{V}$ . This arises naturally, as an ordering of both static and dynamic nodes would not be meaningful, given the construction of the mixed-BN model. A valid parent set, however, can consist of nodes in both  $\mathcal{V}$  and  $\tilde{\mathcal{V}}$ . Therefore, in order to determine the highest scoring parent sets for  $X_i$ , we compute the scores of all the edges  $X_j \rightarrow X_i$ , where  $X_j \in \mathcal{V} \cup \tilde{\mathcal{V}}$ . Then we only consider the  $C$  more promising parents and form the possible parent sets of  $X_i$  as before.

It is proven in ([13]) that order MCMC outperforms structure-MCMC in terms of mixing and convergence of the chain. One substantial advantage of the space of orders is that is is significantly smaller ( $\mathcal{O}(n!)$ ), while, as we previously discussed, the structure space is enormous in high dimensions. Moreover, the posterior over the space of orders is smoother, in contrast to the posterior over the space of structures, which appears to suffer from multiple peaks. It is therefore very common for structure-MCMC to linger around local optima which constitutes to a slower mixing and convergence of the chain. As we will see in our experiments, structure-MCMC was inefficient in more demanding settings (i.e high number of observations in the data), where order-MCMC proved to be superior.

However, order MCMC comes with a disadvantage: If a uniform distribution over the space of orders is imposed, then a non-uniform prior over the space of structures is induced, which cannot be explicitly specified. Networks that are consistent with more orders (for instance, sparser graphs) are more likely than others. For example, the empty network ( a network with no edges) is consistent with all orders and thus can be sampled from any order, whereas a *chain graph*-that is, a network of the form  $X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_N$  is only consistent with one single order. Even equivalent structures can end up having dissimilar priors-if we reverse the first edge in the directed path  $X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_N$ , then we will obtain a graph that is within the same equivalence class, but is more likely a priori, as it is consistent with more than one orders. This bias can cause a problem, especially in smaller domains, where the prior has a stronger impact on the posterior.

## 2.5 Posterior probability of edge relation features

Both structure and order MCMC allow us to obtain a sample  $G_1, G_2, \dots, G_T$  of high scoring structures, which we can use in order to estimate the posterior probability of *edge relation features*. For static and mixed BNs, we need to evaluate the posterior probabilities of the edges over equivalence classes. Recall that, for these two classes of Bayesian networks, an edge sometimes indicates a bidirectional relation. Therefore, for static and mixed BNs we need to extract the CPDAGs of the sampled networks. This is not required for dynamic Bayesian networks, where there are no equivalence classes.

A consistent estimator of the posterior probability of a dynamic edge relation feature is the fraction of structures in the sample in which the edge is present.

$$\hat{P}(f|D) = \frac{1}{T} \sum_{t=1}^T I(G_t, f) \quad (29)$$

where

$$I(G, f) = \begin{cases} 1 & \text{if } f \text{ exists in } G \\ 0 & \text{otherwise} \end{cases}$$

For the static edge posterior probabilities, we use the same estimator, but we average over the CPDAGs of  $G_1, G_2, \dots, G_T$  instead.

When the true underlying network  $G = (\mathcal{V}, \mathcal{E})$  is known, then we can evaluate the network reconstruction accuracy using *receiver-operator-characteristic*(ROC) curves [18]. For various thresholds  $\theta$ , we consider the set  $\mathcal{E}_\theta = \{e \in \mathcal{E} | \hat{P}(e|G) > \theta\}$ . Then, we quantify the number of *true positives*,  $|\mathcal{E}_\theta \cap \mathcal{E}|$ , the number of *false positives*,  $|\mathcal{E}_\theta \setminus \mathcal{E}|$ , the number of *false negatives*, and the number of *true negatives*  $|\mathcal{E}_\theta^c \cap \mathcal{E}^c|$ . A ROC curve is simply a plot of the *sensitivity*, or *true positive rate*,  $(\frac{TP}{TP+FN})$  against the *inverse specificity*  $(\frac{FP}{TN+FP})$ , or *false positive rate*, over all chosen thresholds  $\theta$ .

The performance of the model can be measured if we consider the *Area under the curve* (AUC), or *Area under ROC* (AUROC) value. The AUC value ranges from 0 to 1, and the larger this value is, the better the model performs. An AUROC value that is equal to 1 indicates a perfect prediction, while a random estimator achieves an AUROC value close to 0.5.

Nevertheless, the AUROC is not always suitable in evaluating network reconstruction accuracy, especially when it comes to comparing the mix BN model to another model, for instance, dynamic

BNs. When dynamic-BNs are applied to data generated from a "mixed network" with  $N$  nodes, that is, a network with both static and dynamic edges, then there will be a disagreement between the dimension of the sampled dynamic networks and the true network. In terms of adjacency matrices, every network in the will be represented as an  $N$  times  $N$  adjacency matrix which reflects dynamic relations, while the true mixed network is represented in a  $2N$  by  $N$  matrix, consisting of a static and a dynamic part. Because of this disagreement, which stems from the fact that the dynamic model is intrinsically incapable of learning contemporaneous relations, the performance of the models are not directly comparable in AUROC terms. One obvious, yet suboptimal approach is to take into account the performance of both models on detecting only dynamic relations. However, if we pursue this approach, we do not take the dynamic model's incapacity of inferring contemporaneous relations under our judgement. If we only focus on the dynamic relations the data, then AUROC values for both models will possibly be similar, indicating that both models perform equally well on mixture data, which overestimates the pure dynamic model. In another example, if we want to evaluate the mixed model and the dynamic model on pure dynamic data, focusing only on the dynamic relations in the data will possibly overestimate the mixed model, as we then ignore all the static edges inferred by the mix BN model, even though some of them might have been inferred incorrectly.

For all the reasons mentioned above, we employ another, more intuitive measure of performance, which allows us to compare between models. For a positive integer  $x$ , we consider the  $x$  highest scoring edges and employ *precision*, or *positive predictive rate* ( $\frac{TP}{TP+FP}$ ) among these edges. Precision allows us to investigate whether the mix model lags behind the other two when it comes to learning relations from "pure" data. If the mixed model achieves similar precision values on pure data, as the corresponding pure model, this implies that the mixed model succeeds in detecting dominant relations, and that it does not infer many wrong edges from the data. In addition, when we employ the models on mixture data, precision will indicate if the mixed model outperforms the other two, by achieving higher precision values.

## 2.6 Advanced methods

Now that we understand how we can learn a network structure from data, we continue building towards our inference strategy on the psychometric data set. This data set is formed by four disjunct data sets, corresponding to different clinical stages of psychosis (patient groups). Our goal is to analyze the data by integrating information from all four patient groups, instead of treating each patient group independently.

In this section, we will describe a coupling approach to handle disjunct data sets, that are hypothesized to imply similar interactions between variables, which is a modification of the model developed by Werhli and Husmeier [24]. We will first briefly describe the original model (subsection 2.6.1), and then discuss the modified version that we employ in our experiments, in order to avert any convergence issues we may possibly encounter. This version includes node ordering, and is described in subsection (2.6.2).

### 2.6.1 The Werhli and Husmeier model

In the first part of the paper [24], biological prior knowledge is incorporated into the MCMC inference procedure as a part of a prior distribution over network structures. This concept is then extended into a second scenario, where the goal is to integrate information from  $I$  different data sets ( $\mathcal{D}_1, \dots, \mathcal{D}_I$ ) obtained under different experimental conditions.

There are two obvious approaches to this kind of problem. The first is to perform independent analyses on the data sets, and thus infer  $I$  different networks. Although this analysis will uncover the differences between the data sets, it will possibly suppress the common characteristics between them. A second approach we can pursue is to unify the data sets into one large data set and infer one single network. This approach will bring to light the similarities between the data sets, but will possibly fail to capture the heterogeneity among the data sets.

As a compromise between these two strategies, the model of Werhli and Husmeier describes a mechanism that allows the exchange of information between the data sets. To that end, an *hierarchical Bayesian model* is introduced: Each data set  $D_i$  is associated its own set of hyper parameters  $\beta_i$  and its own network structure  $\mathcal{M}_i$ . An additional, latent structure  $\mathcal{M}^*$  is introduced, which is referred to as the *hyper network*. A prior distribution over the space of valid structures  $\mathcal{M}_i$  is imposed, whose goal is to subtly urge the networks  $\mathcal{M}_i$  to stay similar to the hyper network. Via their bonding to the hyper network, the networks  $\mathcal{M}_i$  are indirectly tied with each other as well.

This prior distribution has the form of a Gibbs distribution:

$$P(\mathcal{M}_i|\beta_i, \mathcal{M}^*) = \frac{e^{-\beta_i|\mathcal{M}_i - \mathcal{M}^*|}}{\sum_{\mathcal{M}_i \in \mathcal{M}} e^{-\beta_i|\mathcal{M}_i - \mathcal{M}^*|}} \quad (30)$$

Borrowing terminology from statistical mechanics, the parameter  $\beta_i$  corresponds to the *inverse temperature*, while the function on the space of structures  $|\mathcal{M}_i - \mathcal{M}^*|$  corresponds to the *energy* of  $\mathcal{M}_i$  and is usually measured in terms of Hamming distance. The denominator is the normalizing constant (partition function).

An MCMC sampling scheme is proposed where all network structures  $\mathcal{M}_i$ , all the hyper parameters  $\beta_i$  and the hyper network  $\mathcal{M}^*$  are sampled from the posterior. We will thoroughly describe the sampling scheme in the next subsection. Werhli and Husmeier compare their proposed coupling scheme to the two extreme approaches, which are referred to as what *monolithic approach*, [24] where the disjunct datasets are merged into one large dataset, and the *uncoupled approach*, where the disjunct data sets are handled independently from one another. Although coupling proves to constantly outperform the other two in various settings, the authors report notable convergence issues in their method. In the next subsection, we discuss how we can attempt to work around this problem by introducing node ordering in the MCMC process, which has shown to accelerate the convergence and mixing of the Markov chain.

### 2.6.2 Introducing node ordering in the Werhli-Husmeier model

In our version of the Werhli-Husmeier model, every data set  $D_i$  is associated with an order,  $\mathcal{O}_i$ . An order can alternatively be seen as a compact representation of the family of its consistent structures. Recall that structure is consistent with an order if it can be formed by choosing independently, for every node, a parent set that is valid given the order. Therefore, the main difference is that instead of a single network structure, every data set is now associated with multiple structures, defined by the order  $\mathcal{O}_i$ . In the original model, the state space of the structures associated with the data is the space of valid DAGs (static Bayesian networks). We can easily generalize the model into the space of mixed BNs.

The modified model can be represented as the hierarchical Bayesian graphical model illustrated in figure (5).

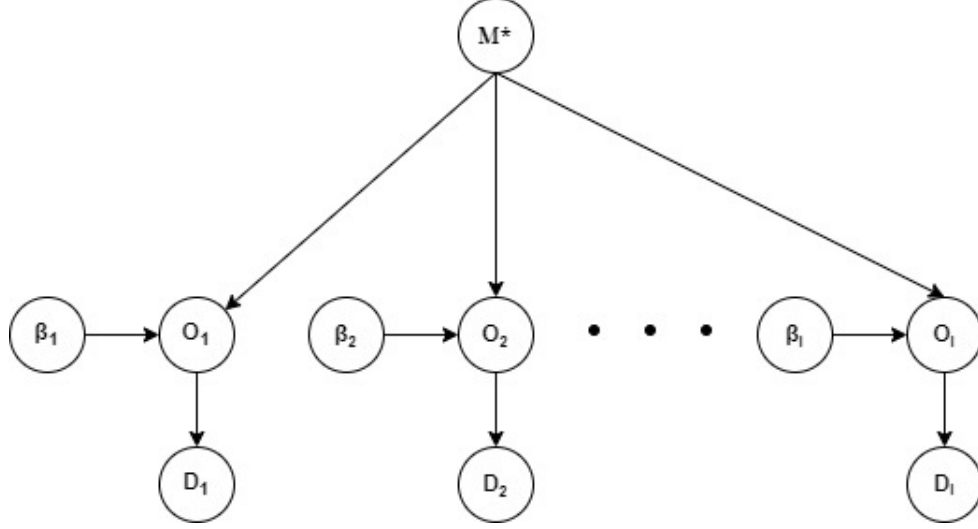


Figure 5: Hierarchical probabilistic graphical model, picture adapted from [24]. Every data set  $D_i$  is associated with a hyperparameter  $\beta_i$  and an order  $o_i$ . The family of networks consistent with the order is binded through a soft constraint to the hyper network  $\mathcal{M}^*$

The local Markov assumption implies that the joint probability of this graphical model is given by:

$$\begin{aligned}
 P(\mathcal{O}_1, \dots, \mathcal{O}_I, D_1, \dots, D_I, \beta_1, \dots, \beta_I, \mathcal{M}^*) \\
 = \prod_{i=1}^I P(D_i | \mathcal{O}_i) P(\mathcal{O}_i | \beta_i, \mathcal{M}^*) P(\beta_i) P(\mathcal{M}^*)
 \end{aligned} \tag{31}$$

For every node  $n$ , we impose a prior over the set of its possible parent sets,  $\pi_n$ . Recall order MCMC inevitably comes with a fan in restriction, which is optional in the original model of Werhli and Husmeier. Therefore,  $\pi_n$  includes only those parent sets with cardinalities that do not exceed the fan in  $k$ . This prior takes the form of a Gibbs distribution over the space of possible parent sets and is given by:

$$P(U, X_n | \beta, \mathcal{M}^*) = \frac{e^{-\beta |U \oplus Pa_{\mathcal{M}^*}(X_n)|}}{Z_n(\beta, \mathcal{M}^*)}, \quad U \in \pi_n \tag{32}$$

where  $Pa_{\mathcal{M}}(X_n)$  represents the parents of  $X_n$  in structure  $\mathcal{M}$  and the operator  $\oplus$  symbolizes the *symmetric difference* between two sets. On a more intuitive level, the energy function quantifies the number of mismatches between a parent set of a node and the corresponding parent set in the hyper network. The denominator  $Z_n(\beta, \mathcal{M}^*)$  is the normalizing constant (partition function) and is equal to:

$$Z_n(\beta, \mathcal{M}^*) = \sum_{U_n \in \pi_n} e^{-\beta |U_n \oplus Pa_{\mathcal{M}^*}(X_n)|} \tag{33}$$

where the sum is over all the possible parent sets  $U_n$  for node  $X_n$ .

In the universe of statistical mechanics, the probability of the system being in a state  $i$  with energy  $E_i$ , among a set of states, each with its corresponding energy, is proportional to  $\exp(-\beta E_i)$ , where  $\beta$  is the inverse temperature. Roughly speaking, at high temperatures and equivalently, low  $\beta$ 's, the exponent drops close to zero, which makes it equally likely for the system to be on any state. In contrast, as temperature drops and  $\beta$  increases, then the exponent is a decreasing function of the energy. In this case, the system discriminates between states, and is more likely to be on states with low energy, rather than states with higher energy.

The analogy is rather straightforward. A lower  $\beta$  imposes a flatter prior over parent sets, which approximates the uniform distribution as  $\beta$  tends to zero. A larger  $\beta$  corresponds to a more meaningful prior, where parent sets with "low energy", that is, parent sets that deviate less from the hyper network, are favoured. Conversely, dissimilarities are penalized, which renders parent sets with "higher energy" less likely to be sampled. Therefore,  $\beta$  can be interpreted as the strength of the coupling between the hyper network and the corresponding order.

Keeping in mind structure modularity, and defining  $Z(\beta, \mathcal{M}) = \prod_n Z_n(\beta, \mathcal{M})$ , we observe that this prior over parent sets induces the following prior distribution over the space of valid structures that are subject to a fan in restriction  $k$ ,  $\mathbb{M}_k$ :

$$\begin{aligned}
P(\mathcal{M}|\beta, \mathcal{M}^*) &= \prod_n P(Pa_{\mathcal{M}}(X_n)|\beta, \mathcal{M}^*) \\
&= \prod_n \frac{e^{-\beta|Pa_{\mathcal{M}}(X_n) \oplus Pa_{\mathcal{M}^*}(X_n)|}}{Z_n(\beta, \mathcal{M}^*)} \\
&= \frac{e^{-\beta|\mathcal{M} - \mathcal{M}^*|}}{Z(\beta, \mathcal{M}^*)} \\
&\approx \frac{e^{-\beta|\mathcal{M} - \mathcal{M}^*|}}{\sum_{\mathcal{M}_n} e^{-\beta|\mathcal{M}_n - \mathcal{M}^*|}}, \quad \mathcal{M} \in \mathbb{M}_k
\end{aligned} \tag{34}$$

where as  $|\mathcal{M} - \mathcal{M}^*| = \sum_i \sum_j |\mathcal{M}_{i,j} - \mathcal{M}_{i,j}^*|$  we denote the Hamming distance between the hyper network and network  $\mathcal{M}$ . In the last equality, we approximate the partition function of the whole network by a product of single-node partition functions  $Z_n(\beta, \mathcal{M}^*)$  (*perfect gas approximation*). The partition function of the whole network is a sum over all possible valid networks  $\mathcal{M}_i$  in  $\mathbb{M}_k$ . As explained in Werhli and Husmeier [24], this approximation slightly overestimates the partition function, as it does not exclude invalid networks, that is, networks with cycles. However, this bias

is a good compromise, as computational complexity is significantly reduced.

The prior over  $\pi_n$  induces a prior over the space of orders as well. Although it cannot be specified, this prior is integrated into the posterior, which is proportional to:

$$\begin{aligned}
P(\mathcal{O}, \beta, \mathcal{M}^* | D) &\propto \prod_n \sum_{U \in \mathcal{U}_{n,o}} \text{score}(X_n, U | D) \cdot P(U, X_n | \beta, \mathcal{M}^*) \\
&= \prod_n \sum_{U \in \mathcal{U}_{n,o}} \frac{e^{-\beta|U \oplus Pa_{\mathcal{M}^*}(X_n)|}}{Z_n(\beta, \mathcal{M}^*)} \text{score}(X_n, U | D), \\
&= \frac{1}{Z(\beta, \mathcal{M}^*)} \prod_n \sum_{U \in \mathcal{U}_{n,o}} e^{-\beta|U \oplus Pa_{\mathcal{M}^*}(X_n)|} \text{score}(X_n, U | D)
\end{aligned} \tag{35}$$

where we borrow the notation [24]  $\text{score}(X_n, U | D)$  to symbolize the local score of node  $X_n$ , given that  $Pa_{\mathcal{M}}(X_n) = U$ . In section 2.3, where we use the notation of [14],  $\Psi[\cdot]$ , it is detailed how the local BGe scores are derived analytically, depending on the model that we employ- static, dynamic or mixed Bayesian networks. The sum in the equalities goes over the set  $\mathcal{U}_{n,o}$ , which represents the set parent sets of node  $n$  that are consistent with the order  $\mathcal{O}$ . We remind the reader that in case of mixed Bayesian networks, an order is a total ordering of the static nodes, while a parent set  $U$  for node  $n$  is consistent with the order if and only if all the static parents in  $U$  precede  $n$  in the order.

## The sampling scheme

Our purpose is to sample the orders, the hyper parameters  $\beta_i$  and the hyper network from the posterior distribution. To do so, we propose new states for our model. We propose new data- associated orders, according to the proposal distribution  $\mathcal{Q}_i(\mathcal{O}_{i_{new}} | \mathcal{O}_{i_{old}})$ , new hyper parameters  $\beta_{i_{new}}$  from the proposal distribution  $\mathcal{R}_i(\beta_{i_{new}} | \beta_{i_{old}})$  and a new hyper network from the distribution The acceptance probability  $A$  is therefore given by:

$$A = \min \left\{ \prod_{i=1}^I \frac{P(D_i, \mathcal{O}_{i_{new}}, \beta_{i_{new}}, \mathcal{M}_{new}^*) \mathcal{Q}_i(\mathcal{O}_{i_{old}} | \mathcal{O}_{i_{new}}) \mathcal{R}_i(\beta_{i_{old}} | \beta_{i_{new}}) \mathcal{W}(\mathcal{M}_{old}^* | \mathcal{M}_{new}^*)}{P(D_i, \mathcal{O}_{i_{old}}, \beta_{i_{old}}, \mathcal{M}_{old}^*) \mathcal{Q}_i(\mathcal{O}_{i_{new}} | \mathcal{O}_{i_{old}}) \mathcal{R}_i(\beta_{i_{new}} | \beta_{i_{old}}) \mathcal{W}(\mathcal{M}_{new}^* | \mathcal{M}_{old}^*)}, 1 \right\}$$

For a new order to be proposed, two nodes are chosen randomly, and they exchange places in the ordering, as explained in [13]. Therefore, the proposal probability  $Q(\mathcal{O}_{i_{new}} | \mathcal{O}_{i_{old}})$  is symmetric. A new hyper parameter  $\beta_{i_{new}}$  is sampled uniformly from the interval  $[\beta_{i_{old}} - d, \beta_{i_{old}} + d]$  for a predefined positive integer  $d$ . In addition, the hyperparameters  $\beta_i$  are not allowed to exceed or

drop below two predefined upper and lower thresholds  $\beta_{max}$  and  $\beta_{min}$ . Therefore the proposal distributions for the hyper parameter  $\beta_i$ ,  $\mathcal{R}_i(\beta_{i_{new}}|\beta_{i_{old}})$  are symmetric as well. If we also consider uniform priors  $P(\beta_i)$ ,  $P(\mathcal{M}^*)$  and symmetric proposal distribution for the hyper network, then, keeping in mind the factorization from 31 the acceptance probability can be rewritten as

$$A = \min \left\{ \prod_{i=1}^I \frac{P(D_i|\mathcal{O}_{i_{new}})P(\mathcal{O}_{i_{new}}|\beta_{i_{new}}, \mathcal{M}_{new}^*)}{P(D_i|\mathcal{O}_{i_{old}})P(\mathcal{O}_{i_{old}}|\beta_{i_{old}}, \mathcal{M}_{old}^*)}, 1 \right\} \quad (36)$$

Werhli and Husmeier propose that sampling is broken into three sub moves, so that acceptance probability is increased. All newly proposed states are accepted according to the Metropolis-Hastings rule, while all the other states are kept unchanged. Firstly, new orders  $\mathcal{O}_i$  are proposed for all data sets in turn, while the rest of the parameters (the hyper parameters  $\beta$  and the hyper network) are fixed. According to the Metropolis-Hastings update rule, the newly proposed order  $\mathcal{O}_{i_{new}}$  is accepted with probability:

$$\begin{aligned} A(\mathcal{O}_{i_{new}}|\mathcal{O}_{i_{old}}) &= \min \left\{ \frac{P(\mathcal{O}_{i_{new}}, \beta_i, \mathcal{M}^*|D)}{P(\mathcal{O}_{i_{old}}, \beta_i, \mathcal{M}^*|D)}, 1 \right\} \\ &= \min \left\{ \prod_n \frac{\sum_{U \in \mathcal{U}_n, \mathcal{O}_{i_{new}}} e^{-\beta|U \oplus Pa_{\mathcal{M}^*}(X_n)|} \text{score}(X_n, U|D)}{\sum_{U' \in \mathcal{U}_n, \mathcal{O}_{i_{old}}} e^{-\beta|U' \oplus Pa_{\mathcal{M}^*}(X_n)|} \text{score}(X_n, U'|D)}, 1 \right\} \end{aligned} \quad (37)$$

Where the product in the last equality goes over all nodes, while the sums go over different parent sets, the ones that are consistent with  $\mathcal{O}_{i_{new}}$  in the numerator, and the ones that are consistent with  $\mathcal{O}_{i_{old}}$  in the denominator. Recall that a parent set for node  $j$  is considered valid given an order, if and only if it is formed by nodes which precede  $j$  in the order. In the newly proposed order, the nodes whose place in the ordering is before and after the flipped nodes keep the same valid parent sets, because they are preceded by exactly the same nodes as in the old order. Therefore, the terms in the product associated with those nodes cross out, and we can focus the computation of the acceptance probability of the new order only on those nodes that lie between the flipped nodes.

Subsequently, new hyper parameters  $\beta$  are proposed for all data sets. Their acceptance probability

is given by :

$$\begin{aligned}
A(\beta_{i_{\text{new}}}|\beta_{i_{\text{old}}}) &= \min \left\{ \frac{P(\mathcal{O}_i, \beta_{i_{\text{new}}}, \mathcal{M}^*|D)}{P(\mathcal{O}_i, \beta_{i_{\text{old}}}, \mathcal{M}^*|D)}, 1 \right\} \\
&= \min \left\{ \frac{Z(\beta_{i_{\text{old}}}, \mathcal{M}^*)}{Z(\beta_{i_{\text{new}}}, \mathcal{M}^*)} \prod_n \frac{\sum_{U \in \mathcal{U}_{n, \mathcal{O}_i}} e^{-\beta_{i_{\text{new}}} |U \oplus Pa_{\mathcal{M}^*}(X_n)|} \text{score}(X_n, U|D)}{\sum_{U \in \mathcal{U}_{n, \mathcal{O}_i}} e^{-\beta_{i_{\text{old}}} |U \oplus Pa_{\mathcal{M}^*}(X_n)|} \text{score}(X_n, U|D)}, 1 \right\}
\end{aligned} \tag{38}$$

where both sums go over the same parent sets, but the change of hyper parameter results in a change of the weights of their scores.

Finally, a transition to a new neighbour hyper network is proposed through either an edge addition , an edge removal or an edge reversal. We do not require the hyper network to comply with the constraints of a mixed network, that is, cycles among static edges are not forbidden. Moreover, it does not need to obey to the fan-in constraint. The probability of accepting the new hypernetwork is equal to:

$$\begin{aligned}
A(\mathcal{M}_{\text{new}}^*|\mathcal{M}_{\text{old}}^*) &= \min \left\{ \prod_{i=1}^I \frac{P(\mathcal{O}_i, \beta_i, \mathcal{M}_{\text{new}}^*|D)}{P(\mathcal{O}_i, \beta_i, \mathcal{M}_{\text{old}}^*|D)}, 1 \right\} \\
&= \min \left\{ \frac{Z(\beta_i, \mathcal{M}_{\text{old}}^*)}{Z(\beta_i, \mathcal{M}_{\text{new}}^*)} \prod_n \frac{\sum_{U \in \mathcal{U}_{n, \mathcal{O}_i}} e^{-\beta_i |U \oplus Pa_{\mathcal{M}_{\text{new}}^*}(X_n)|} \text{score}(X_n, U|D)}{\sum_{U \in \mathcal{U}_{n, \mathcal{O}_i}} e^{-\beta_i |U \oplus Pa_{\mathcal{M}_{\text{old}}^*}(X_n)|} \text{score}(X_n, U|D)}, 1 \right\}
\end{aligned} \tag{39}$$

We note that the likelihood of a parent set given the data is not affected by a change of the hyper network  $\mathcal{M}^*$  or the new hyper parameters  $\beta$ . Therefore, the acceptance probability of these states strictly depends on the prior terms for the parent sets . If the current order,  $\mathcal{O}_i$  allows for structures that are more similar to the newly proposed hyper network, then the acceptance probability in (39) increases.

Let us now consider the sampling of the hyper parameters  $\beta$ . If the symmetric differences in the exponents in (38) are systematically close to zero, then, regardless of the value of  $\beta$ , the exponential terms will be suppressed, and the numerator will become approximately equal to the denominator in the terms of the product. The acceptance probability will then only depend on the normalization constant, which is, as we previously mentioned, decreasing in  $\beta$ . Consequently, if the structures consistent with the order tend to be similar to the hyper network, then higher values

for  $\beta$  are favoured in the sampling process. In contrast, if they tend to deviate from the hyper network, then  $\beta_i$  is more likely to linger in lower intervals, since a larger  $\beta_i$  will lead to a more significant shrinkage of the exponential terms, given that the symmetric differences in the exponents are rather large.

### 3 Data

#### 3.1 The RAF signalling pathway

The RAF signalling pathway ([22]) is a well-known gene regulatory network, which describes the information flow within a cell. The network can be seen in figure 6. The nodes of the network represent proteins, while the edges encode the cascade of *signal transduction*.

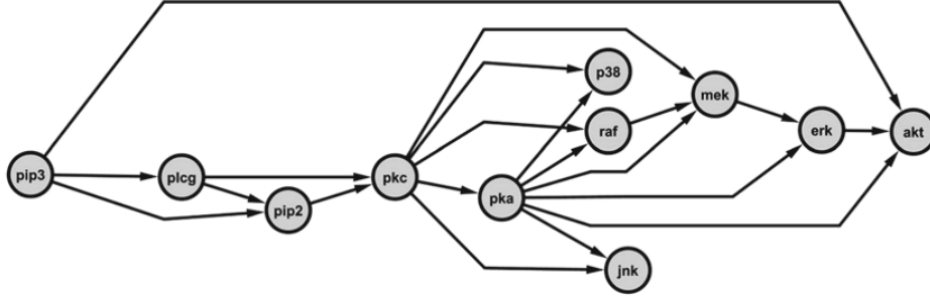


Figure 6: The Raf signalling pathway, picture from [14]

Under certain circumstances, information flow is stimulated inside a cell. The molecules that are involved respond to the information they receive, which affects subsequent molecules in the pathway. The proteins are activated through a process called *phosphorylation*, which is regulated by a special group of enzymes, *protein kinases*. Protein kinases modify the proteins in the signalling cascade by attaching a phosphate group to them. This event impacts the functionality of the proteins in the cascade, which triggers chemical reactions that subsequently affect the neighbouring molecules in the pathway.

Protein kinases are known to orchestrate the majority of cellular pathways. The malfunction or imbalanced activity of the kinases can lead to diseases, such as cancer.

Using the RAF signalling pathway as a gold standard network, we can generate synthetic data and evaluate the network reconstruction accuracy of our methods. In order to generate independent state-space observations (pure static data), the observation of variable  $X_i$  is equal to:

$$X_i = \sum_j \beta_{ji} X_j + \epsilon_i$$

where  $\beta_{ji}$  is the strength corresponding to the edge  $e_{ji}$ , and  $\epsilon_i$  is an  $m$  dimensional vector that represents noise, usually sampled from the standard normal distribution. In case there is no edge from

node  $j$  to node  $i$ , then  $\beta_{ji} = 0$ . As a result, the data matrix expands with respect to the topological order of the network.

We can generate pure temporal data from the RAF signalling pathway in a similar manner. Here, every node is influenced by its parents with a time delay  $\tau$ . Having assumed that  $\tau = 1$ , we can generate the realization of node  $i$  at time point  $t$  from:

$$X_i(t+1) = \sum_j \beta_{ji} X_j(t) + \epsilon_i(t+1)$$

Here, the data matrix expands column-wise, and the topological order of the network does not need to be taken into account.

Finally, in order to generate mixture data, the edges of the gold-standard network were divided into two groups, one corresponding to the static, and the other corresponding to the dynamic edges. Then, the realization of node  $i$  at time point  $t$  is given by:

$$X_i(t) = \sum_j \beta_{ji} X_j(t) + \tilde{\beta}_{ji} \tilde{X}_j(t-1)$$

where  $\beta_{ji}$  is the strength corresponding to the static edge  $e_{ji}$ , while  $\tilde{\beta}_{ji}$  corresponds to the dynamic edge  $\tilde{e}_{ji}$ . Again, we need to take into account the ordering of the nodes.

### 3.2 Psychometric data

The *Mapping Individual Routes of Risk and Resilience(Mirror) study* ([5]) is an innovative study whose main objective is to investigate dynamic networks as means of predicting the progression of psychotic symptoms. The study adopts a network-based approach to psychopathology, which is based on the hypothesis that mental disorders result from interplay between symptoms ([6]). Most currently established "disease models" view symptoms as independent expressions of an underlying disorder. In contrast to these models, the network approach conceptualizes mental disorders as an causally interconnected system of symptoms (*symptom networks*). From that perspective, symptoms are not independent from one another, they interact through causal relations over time ([5]).

The data set consists of 90 measurements for 96 patients aged between 18-35, with ranging psychopathology. The patients are allocated into 4 different groups, which represent successive clinical stages: Patients of group 1 express milder symptoms, while patients of the last group show

graver symptoms and are in high risk for psychotic disorder. The measurements focus on 38 different symptoms.

Given the patient specific data sets, we constructed four group specific data sets, that is, all data sets corresponding to the patients of a specific group were merged into one, large data set. We observed a significant number of missing measurements, which we imputed using a moving average process with window size  $K=4$  and exponential weights: A missing measurement was thus replaced taking into account the four consecutive values before and after it, multiplied with weights that decrease exponentially:

$$X_i(t) = \sum_{s=1}^4 \left(\frac{1}{2}\right)^s X_i(t-s)$$

It is expected that if the four group-specific data sets are treated independently, then four different networks will be inferred. However, as we intend to capture the continuum of psychosis by highlighting both the heterogeneity and similarities between different clinical stages, we introduce coupling to the analysis of the psychometric data by employing the modified version of the Werhli and Husmeier model as described in section (2.6.2) .

## 4 Implementation details

In this section we provide a brief description of the experiments we performed using the methods previously discussed under section 2. All implementations were in MATLAB, and we also made use of the Peregrine cluster. In the first subsection, we present some details on the evaluation process of our inference methods on synthetic Gaussian data. These experiments were conducted in order to confirm that our inference methods were correct, before proceeding to further analyses on real world data. In the second subsection, we give a description of our approach on the psychometric data (4.2). For more details on the software we developed for our implementation, we refer the reader to the Appendix, at the end of the document. There, we discuss the main functions that we created for our analysis. A zip folder with the code has been provided to my supervisors, and is available upon request.

### 4.1 Synthetic data

#### 4.1.1 Evaluation of the mixed Bayesian network model

In our MCMC simulations, our main objective is to evaluate the mix model in various settings, and to compare its performance to the already established 'pure' models, with respect to network reconstruction. For this purpose, we run structure MCMC simulations on synthetic data, generated from the RAF signalling pathway ( $N=11$ ), as described in section 3.1. The edge strengths  $\beta_{i,j}$  were sampled uniformly from within the interval  $[0.5, 2]$ , and then weights with the negative sign with probability  $1/2$ . In all our simulations, the generated data is standardized to zero mean and marginal variance of one, for all variables. We have tuned the priors so that they have the minimal possible impact on the posterior: The prior distribution over the space of structures is the uniform prior, while the hyperparameters of the Wishart prior are tuned to the most uninformative values possible:  $\alpha = N + 2$ ,  $v = 1$ ,  $T_0 = 0.5 \mathbb{I}_N$ ,  $\mu_0 = (0, 0, 0, \dots, 0) \in \mathbb{R}^{1 \times N}$ . For the dynamic structure MCMC simulation,  $N$  is substituted by  $N + 1$ , as self loops were allowed. Similarly, for the mix structure MCMC simulation,  $N$  is replaced by  $2N$ . We defined the initial state of all simulations to be the empty graph, that is, the graph with no edges. This is expressed as an all-zero square matrix with  $N$  rows in case of the pure models, and as an all zero matrix of  $2N$  rows and  $N$  columns for the structure mix-BN simulation.

In order to ensure that sufficient convergence was achieved, we run independent MCMC simulations, with different initializations for the first state/graph, and monitored the number of iterations by plotting the resulting edge scores. Every simulation consisted of 500000 iterations, the first

100000 of which was discarded (burn-in phase). The frequency of sampling is every 1000 steps. First, we evaluate the performance of the mix-BN model using AUROC values, and then, we move on to comparing the models in different settings. For static observational data, we compare the precision with respect to the highest scoring  $x = 20$  edges, of the static and the mix BN models. For the pure dynamic data, we compare the precision with respect to the highest scoring  $x = 20$  edges, of the dynamic and the mix BN models. For mixture data, we compare the models using the same metric.

#### 4.1.2 Comparison between order and structure MCMC

In our order MCMC simulations, all hyper parameters for the BGe model were specified as described in the previous subsection, while the initial order is taken to be a random permutation of the nodes. Moreover, we impose a fan in restriction of 3. The parameter  $C$  which corresponds to the number of parents that will be involved in the forming of all possible families for every node, is set to  $C = 10$ . We run all simulations for 5000 iterations, discarded the first 2000 states in the trajectory, and sampled every 100 steps.

In order to illustrate how node ordering can overcome the convergence issues that come with structure MCMC, we compare the two models in terms of convergence. We run the simulations on synthetic data, generated from a disjunct union of 3 copies of the gold standard network depicted in 6 (N=33). We apply the models on various data sets, where the changing factor is the number of observations. We adjust the number of iterations accordingly. All data sets were standardized to zero mean and variance equal to 1, for all nodes.

#### 4.1.3 The coupling scheme

We will evaluate our modified version of the Werhli-Husmeier model ([24]) on mixture data. We fix a gold-standard network and generate three data sets. Therefore, all three data sets have the same underlying network, but the edge strengths differ. We include a fourth data set, which consists of pure noise. Our objective is to compare AUC values between this approach and the two extreme approaches presented by the authors, that is, the *uncoupled* and the *monolithic* approach, in order to investigate whether the coupling approach leads to any improvement when it comes to network reconstruction.

Moreover, we are interested in evaluating to what extent the hyper parameters inferred by our

model reflect the true relation between the data sets and the hyper network. Recall that  $\beta$  indicates the strength of the coupling between a data- associated order and the hyper network. Therefore, the hyper parameters associated with the first three, 'true' data sets are expected to reach higher values on average, in comparison to the fourth data set, which consists of nothing but noise. The hyper parameter associated with this data set should move in lower intervals, indicating the absence of a meaningful prior over the state of orders: No order explains this data set significantly better than any other. In other words, the question is whether our model is able to identify the "corrupted" data set, among the true data sets.

The initial hyper parameters for all data sets are uniformly sampled from within the interval  $[0, 30]$ . In every step of the algorithm, new hyper parameters  $\beta_{i_{new}}$  are sampled uniformly from a window of length  $d=4$ , centered at the current value of  $\beta_i$ . The hyper parameter  $\beta_i$  is also forced to stay within the interval  $[0, 30]$ .

Finally, an initialization for the hyper network is required. Werhli and Husmeier ([24]) mention that the empty network is not an optimal initial state, as it will give a significant lead to the corrupted data set. For our initialization, we run a short order-MCMC simulation on the unified ('monolithic') data set. Once we obtain a sample of networks, we average over them and select the only the highest scoring edges to be present in our initial hyper network. The initial orders associated with the four data sets were random permutations of the nodes. We run the model for  $10^4$  simulations, discarded the first  $5 \cdot 10^3$  states and sampled every 100 steps.

## 4.2 Psychometric Data

As a first approach to handling the psychometric data set, we employed structure MCMC inference methods. Nevertheless, because of the dimensionality of the four datasets, structure MCMC never reached convergence, which drove us towards order MCMC. Our initial pursuit was to confirm that order MCMC can reach convergence.

In order to apply order MCMC, we need to sensibly fixate the number  $C$  of the highest scoring parents that will be considered as potential parents of the node accross the simulation. It has been previously discussed how this number can have a paramount impact on the success of the algorithm and that the inclusion of such an approximation is not optimal. However, performing a full enumeration of all possible parent sets for a node led to immense computational costs, given the dimensionality of the data sets-recall the mix-BN model doubles the nodes of the network. Therefore, we settled for  $C = 20$ . self loops were considered valid edges in these settings. Our experiments led to the conclusion that a sufficient amount of convergence is reached at  $5 \cdot 10^4$  iterations. The first  $2 \cdot 10^4$  states were discarded (burn-in phase) and the frequency of sampling was every 100 steps.

As a first approach to the data, we performed a simple "leave-one-out" cross validation scheme on the 96 patients, in order to compare between the Bayesian network models. Our goal is to approximate the probabilities  $P(D_{patient}|D_i)_{i=1,..,4}$ , where  $D_{patient}$  represents the 38 times 90 patient specific data set, and  $D_i$  represents the  $i$ -th group specific data set, which we consider as a "prior" data set which we can use to specify the hyper parameters. The part of the data associated with the patient has firstly been removed from the corresponding group-specific data. Then, an order MCMC simulation is run on the 4 group specific data sets, and a sample of networks  $G_1, G_2, \dots, G_T$  was obtained for each one of them. The predictive probability  $P(D_{patient}|D_i)$  can be approximated by:

$$P(D_{patient}|D_i) = \frac{1}{T} \sum_{t=1}^T P(D_{patient}|G_t)$$

When the predictive probability  $P(D_{patient}|G_t)$ , the hyper parameters for the BGe model are not suppressed like in the previous experiments.  $T_0$  is taken equal to the  $T_{D,m}$  matrix, computed on the prior data set  $D_i$  (20, 12, ??). The hyper parameters  $a$  and  $v$  are set equal to the second dimension of the prior data set. The group specific data sets were standardized, but the patient-specific data sets were left unstandardized.

Our goal is to classify each patient, to the group that maximized the predictive probability  $P(D_{patient}|D_i)$ , in other words, to the group that best explains the patient-specific data. We do so with all three Bayesian network models: A sample of static, dynamic and mixed Bayesian network is gathered from the group specific data sets. The goal of this experiment is to compare the missclassification rates between models. This is a very expensive approach, as for every patient, we need to run three new order MCMC simulations on the data of the group the patient is allocated to, after extracting the part of the data that is associated with the specific patient.

Finally, we applied the modified version of the Werhli and Husmeier model on the four data sets. The model was run for  $10^5$  iterations, the first  $5 \cdot 10^4$  states were discarded and we sampled every 100 steps. In order to facilitate convergence, we used more informative initializations for the states: We run greedy search algorithms on the four data sets, to determine one high scoring network for each group. The hyper network was chosen as the consensus of these networks. The topological orders of these networks were chosen as the initial states for the groups. The hyper parameters  $\beta_i$  were initialized as described above.

## 5 Results

### 5.1 Evaluation of structure mix BN

As a first step, we tested the mix-BN model on synthetic mixture data ( $n=11$ ) generated from the RAF signalling pathway under various conditions and evaluated its performance using AUROCs. In this experiment, we vary the number of observations  $m$ . In figure 7 it appears that the model achieves high AUROCs, which indicates a good performance.

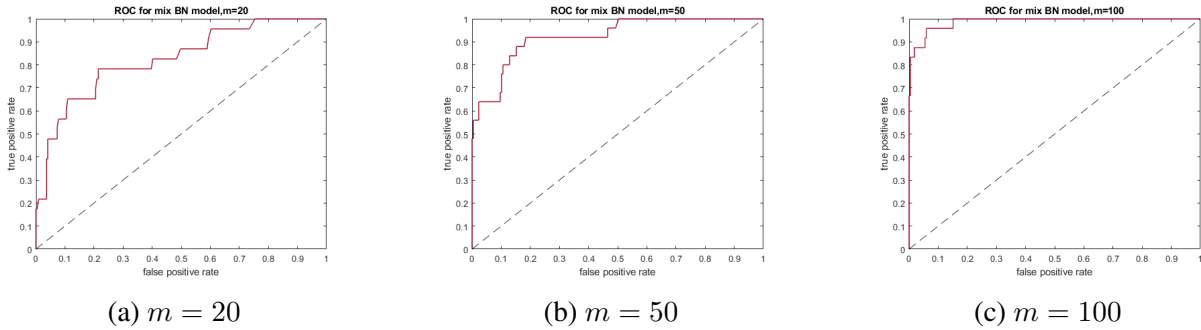


Figure 7: ROC curves

The AUROCs seems to drop with  $m$ . This is only reasonable, since the mix model doubles the number of potential parents. Especially for  $m = 20$ , the number of predictors exceeds the number of observations, which accounts for lower AUROC values.

As a next experiment, we want to compare the performance of the mix-BN model to the performance of the pure models, on synthetic data. As we have previously discussed under 2.5, AUROCs are not comparable between models, and therefore, as our evaluation criterion we employ the precision among the  $x=20$  highest estimations for the edge scores.

We first evaluate the models on mixture data. In figure 8, we illustrate bar plots of average precision values achieved by all models, over 10 independent simulations on different data sets consisting of mixture data. We observe that the mix-BN models follows the expected trend, with sufficiently high precision values, which, similar to AUROCs, drop with the number of observations. In addition, the pure models clearly lag behind, with precision values systematically bounded by 0.5. This also does not come as a surprise: The pure static model is intrinsically incapable of inferring dynamic edges (and vice versa), and as we will shortly discuss, can potentially infer wrong edges.

Hence, we expected the pure models to learn at most half of the true edges.

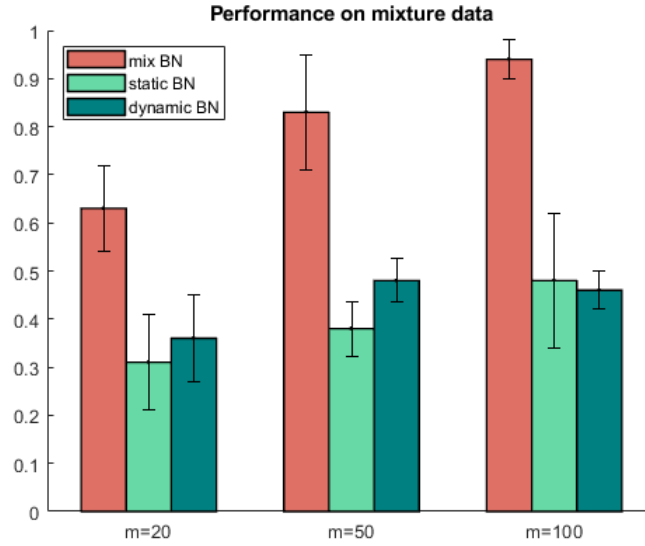


Figure 8: Comparison of precision amongst  $x = 20$  highest scoring edges between models

We are particularly interested in investigating if the new model is capable of detecting dominant features on pure data as well as the pure models. To this end, we now evaluate the models on pure static and dynamic data. On pure data, the challenge for the mix BN model is to restrict in only learning the true edges and not additional, irrelevant edges.

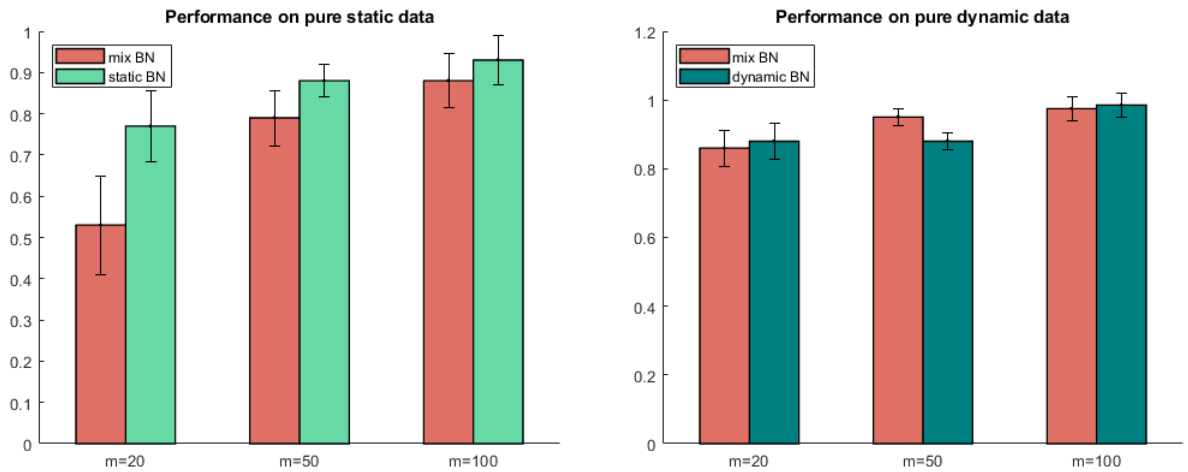


Figure 9: Performance on pure static (left panel) and pure dynamic data(right panel)

In figure 9, we illustrate bar plots of average precision values, over 10 simulations on 10 independently generated data sets, consisting of observational static (left panel) and time series (right panel) data. On the right panel of figure 9, we observe that the mix BN models performs just as well as the pure dynamic models.

On the left panel of figure 9, where the performance of mixed and static BN are evaluated on pure static data, we see that precision values for the mix BN model are not far below compared to those of the static BN. It can however be seen that the gap between the mixed and the pure model on pure static data is wider than it is on pure dynamic data. We suspect that this happens because mix-BN misidentifies the directions of some of the edges. This stems from the fact that some dynamic edges are probably accepted in the beginning of the simulation, which interferes with the extraction of the CPDAG. For instance, a true, bidirectional edge can be classified as compelled by the mix BN model, because of the presence of a dynamic edge. To test our suspicion, we ignore the directionality of the static edges and calculate precision values under the consideration that all static edges are bidirectional. The results can be seen in figure 10:

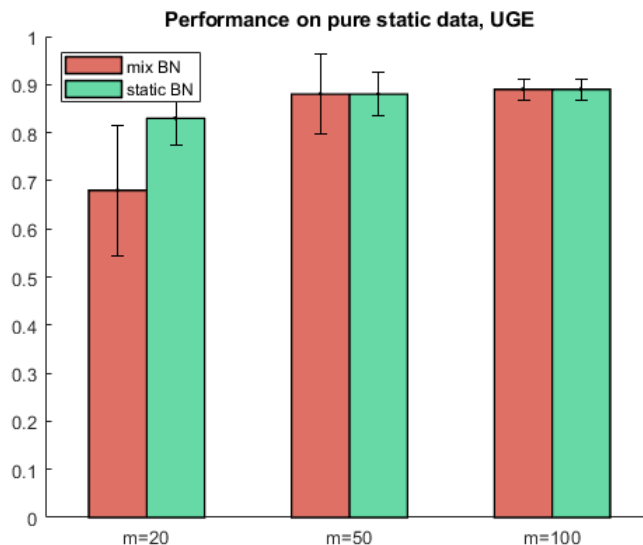


Figure 10: Comparison of precision amongst  $x = 20$  highest scoring edges between mix BN and static BN on pure static data, without taking directionality into account

In this setting we observe a significant improvement in precision values for the mix BN model, as it now seems to perform equally well as the static BN model. However, it clearly lags behind when the number of observations is low (relative to the number of variables).

After comparing precision values, we further investigate the performances between models. More precisely, we want to examine whether the mix BN model gives similar estimates for pure edges, as the pure models, when applied on pure data. To that end, we create scatter plots of the estimates of the edge scores between models, obtained from simulations on pure data. In figure 11 we illustrate the estimates for the true edges obtained from one simulation of the mix BN and the dynamic BN models on pure dynamic data.

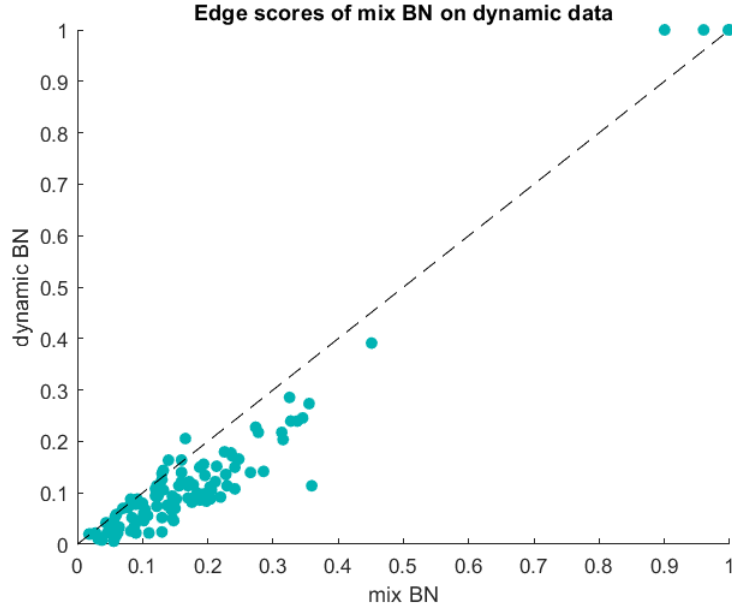


Figure 11: Edge scores of mix BN versus edge scores of dynamic BN on pure dynamic data.

This scatterplot indicates that the mix BN assigns very similar scores on dynamic edges as the dynamic BN model.

In the case of pure static data, we do not expect such strong agreement between models, as our experiment so far have indicated a small, yet considerable deviation in results. Indeed, the left panel of figure 12, where estimates for the edge scores obtained by mix and static BN on pure static data are compared, confirms our presumption: It appears that the mix BN not only overlooks some strong edges, it also mistakes some weak relations as rather dominant. However, the right panel provides evidence that , if we ignore the direction of the static edges, then the edges that are inferred incorrectly by the mixed model reduce by a significant fraction.

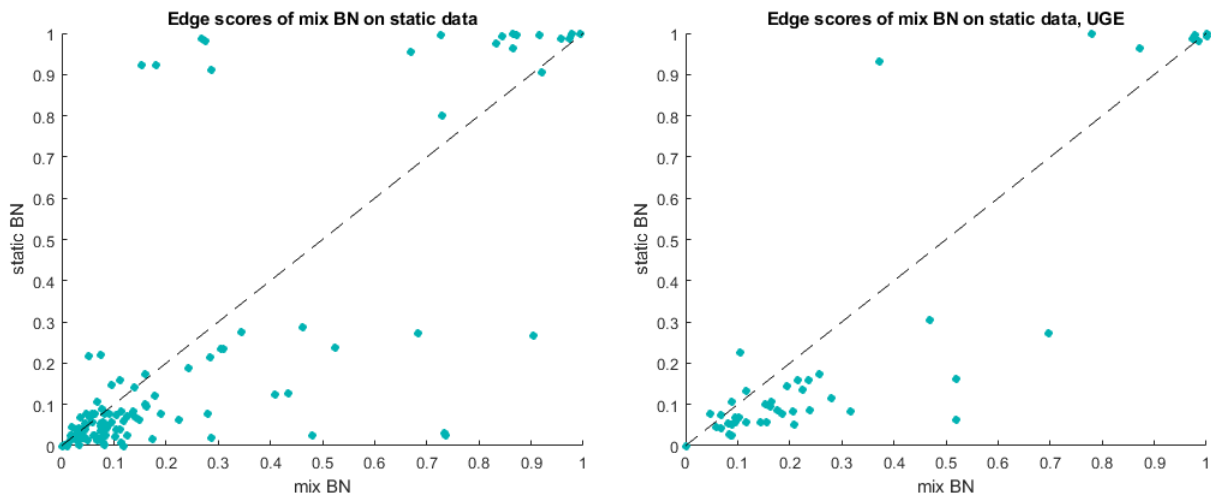


Figure 12: Edge scores of mix BN vs static BN on pure static data when edge directions are considered (left panel) and ignored (right panel).

However, there is still some disagreement between models, which mostly stems from the fact that the mix BN model seems to learn wrong static edges. A possible explanation for this is that some interactions that are mediated by other variables are identified as direct interactions by the mix BN model. More specifically, if two nodes share the same dynamic parent, then they have an indirect contemporaneous interaction. In this scenario, the mix model can potentially infer a static edge that is not among the true edges.

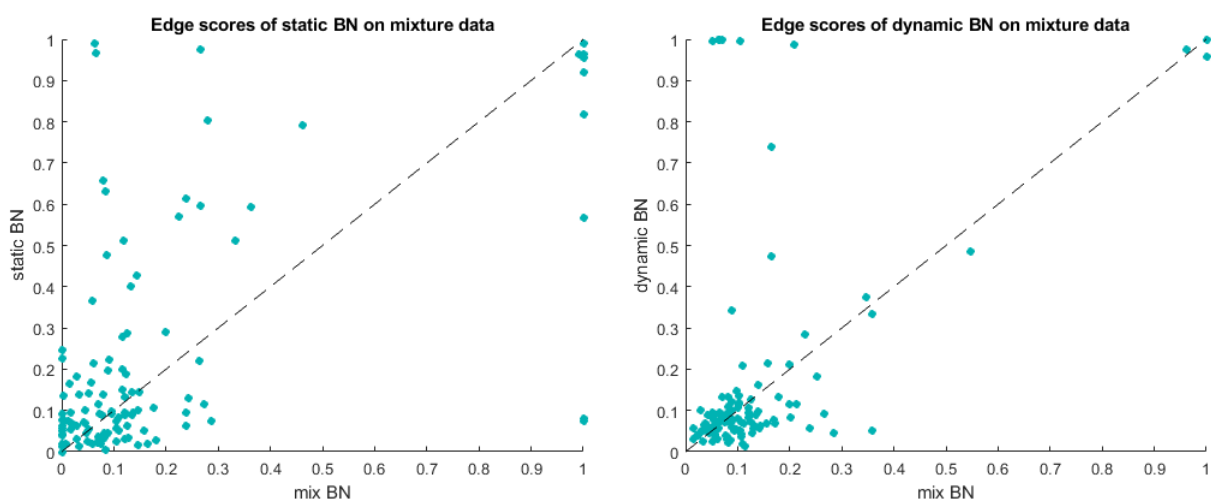


Figure 13: Edge scores of mix BN versus static (left panel) and dynamic (right panel) BN on mixture data.

This is one of the challenges that the static BN too has to face when employed on mixture data. The static model cannot see dynamic edges, but it might be misguided by their effect and either learn wrong static edges, or mistake compelled edges for bidirectional- if a static edge participates in a mixed v-structure, then , from a static point of view, it will be classified as reversible. Figure 13 illustrates the results of the previous experiment, from simulations on mixture data, mix versus static (left hand side) and mix versus dynamic (right hand side). As suspected, the scatterplot on the left hand side of 13 shows a strong disagreement between models, as it seems that many wrong edges are inferred by the pure model, while some true static edges are overlooked. The results on dynamic edges (right hand side of figure 13 show that the dynamic model does not miss true dynamic edges, but it appears that it learns some wrong ones, possibly because indirect interactions are, also in this case, misinterpreted for direct ones: If node  $A$  has an incoming dynamic parent  $B$  and an outgoing static edge pointing to node  $C$ , then there is an indirect dynamic interaction between nodes  $C$  and  $B$ , mediated by  $A$ . Although the static edge cannot be seen by the dynamic model, it can possibly detect this interaction and interpret it as an edge.

## 5.2 Order MCMC

After detecting significant convergence issues in the structure MCMC approach, especially as dimensionality increases, we investigated whether convergence and mixing of the Markov chain can be improved by employing order MCMC instead ([13]). Given that the data sets we intend to analyze are rather large ( $n = 38$ ,  $m \approx 2000$ ) we compare structure and order MCMC on a dataset generated by the disjunct union of smaller mixed networks. The resulting mixture data sets have  $n = 33$  nodes, and the  $m$  ranges. Below we compare the two MCMC approaches in terms of convergence, by the use of scatterplots.

On the right hand side of figure 16, we see the scatter plots of the edge scores, on three independent data sets with  $m = 100$ ,  $m = 500$  and  $m = 1000$ . Iterations were increased with  $m$ , namely, we run the simulations for  $5 \cdot 10^5$ ,  $10^6$  and  $5 \cdot 10^6$  steps respectively. From a rather low number of observations, we see that structure MCMC comes with notable convergence issues. As observations increase, convergence becomes more and more unreliable.

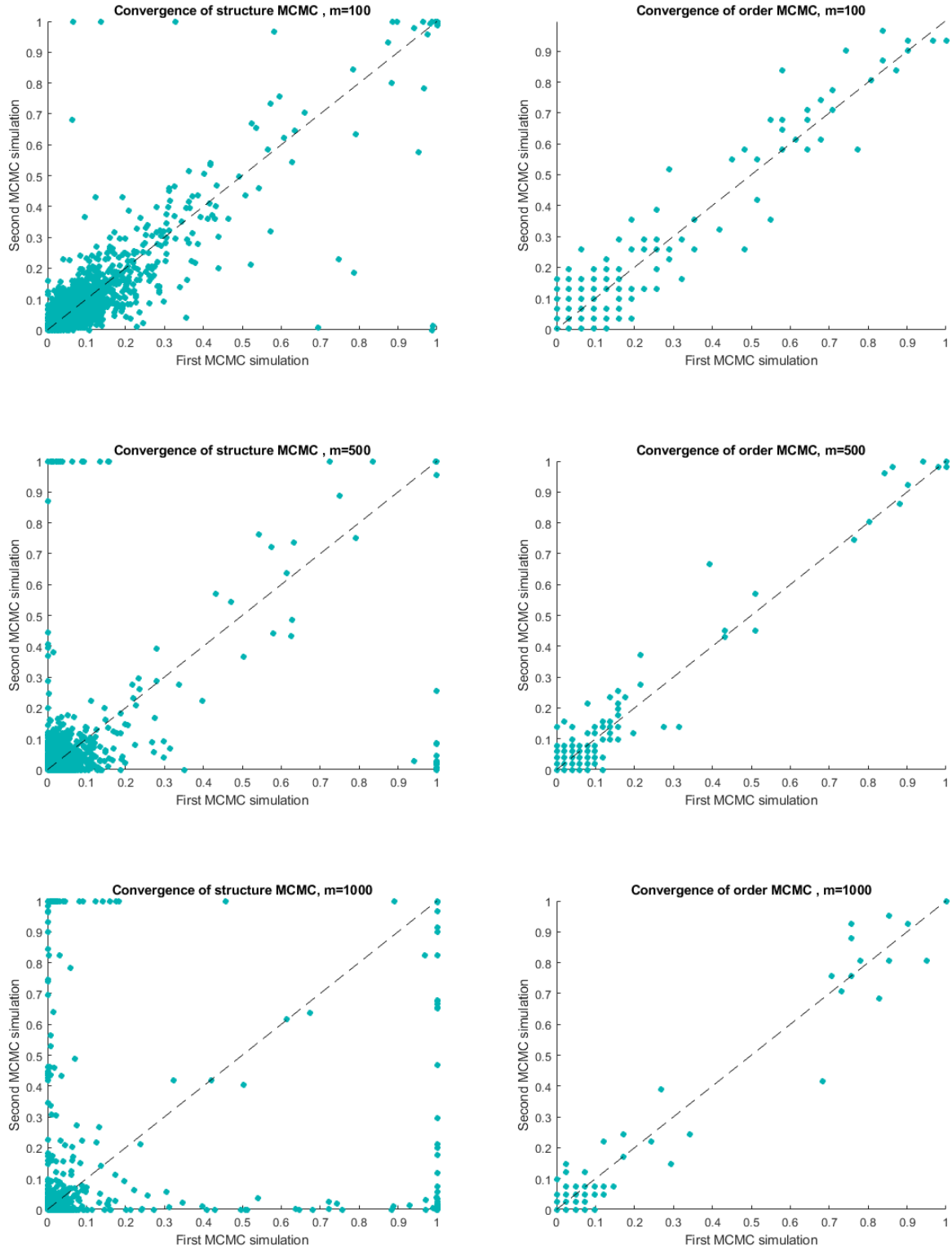


Figure 16: Convergence of structure (left side) and order (right side) MCMC on synthetic data, with  $n = 33$  and increasing  $m$ . The first row corresponds to  $m = 100$ , the second to  $m = 500$  and the third to  $m = 1000$ .

Whereas the number of observations seems to have a strong effect on the convergence of the Markov chain in structure MCMC , it seems to have no impact whatsoever on the convergence of order-MCMC. This can be seen on the right hand side of the figure. As the number of observations increase, we leave the number of iterations unchanged and equal to  $6 \cdot 10^3$ . In all three settings, convergence is achieved early, compared to structure MCMC. Convergence of structure and order MCMC were previously discussed on section, and this experiment confirms that order MCMC is preferable when it comes to larger data sets.

### 5.3 Integration of data sets derived under different conditions

In this section, we illustrate the empirical results on the modified version of the model proposed by Werhli and Husmeier. We generate three data sets from the same gold standard network, but with different parameters, namely, different regression coefficients. We will refer to those data sets as the "true" data sets. In the first experiment, we compare the performance of the model to the two "extreme approaches". In the first approach, we handle the three data sets independently, whereas in the second, we merge the three data sets into one single, large data set. Borrowing terminology from ([24]), we refer to the three approaches as the "coupled", the "uncoupled" and the "monolithic" approach. As our evaluation criteria, we employ AUROC and precision among the  $x = 20$  highest scoring edges.

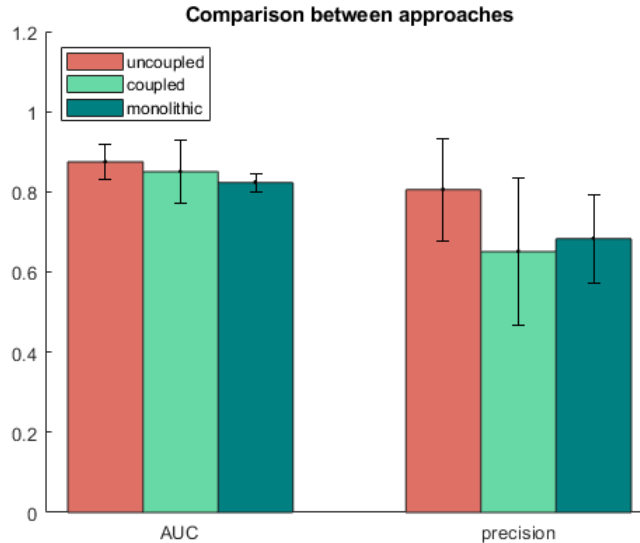


Figure 17: Comparison of AUROC and precision values between our coupling scheme, and the two extreme approaches.

We expect to see that the coupling improves performance. Weak underlying relations between variables, that is, regression coefficients of small magnitude are usually harder to detect, which leads to lower AUROC/precision values. However, under the light of more data sets, that are generated from the same network, we expect those more subtle interactions to be more easily detected. On figure 17, we compare the average AUROC and precision values over 5 independent runs of the three schemes on different data. It can be seen that coupling indeed leads to both higher AUROC and higher precision values.

For the second experiment, we introduce a fourth data set, that consists of pure noise. We will refer to this data set as the "corrupt" data set.

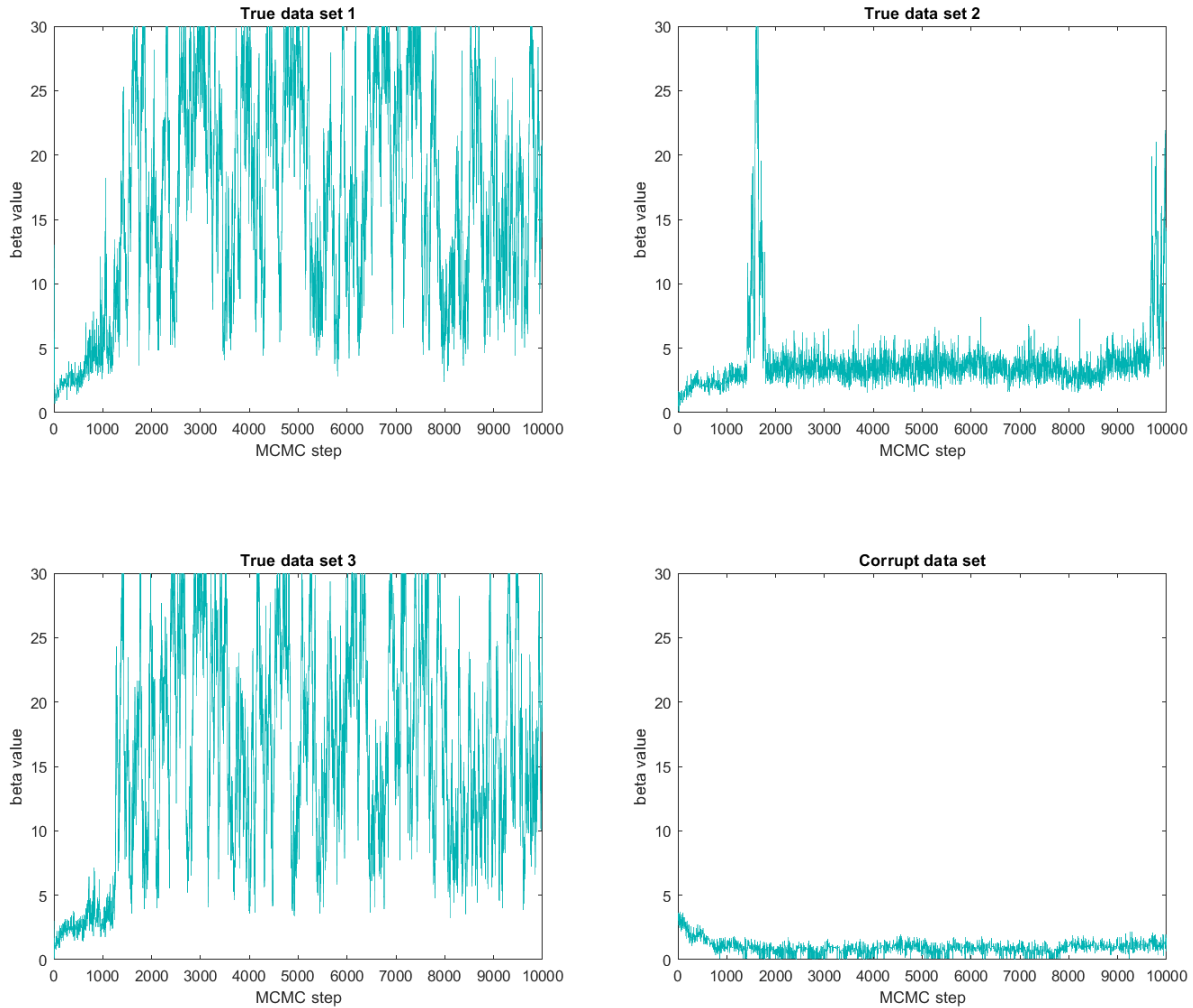


Figure 19: Traceplots of the hyperparameters beta. Larger Beta values indicate a stonger coupling with the hypernetwork. The lower right panel corresponds to a data sets that consists of pure noise, while the other three corresponds to 3 "true" datasets, generated by the same gold standard network, but with different parameters.

Recall that in this scheme, every data set is associated with a hyper parameter  $\beta_i$ , which, as we discussed in section 2.6.2, indicates the strength of the coupling between the order/ family of networks associated with dataset  $D_i$ , and the hyper network. We expect stronger coupling (larger  $\beta$  values between the orders associated with the true data sets and the hyper network , in comparison to the order associated with the corrupt data set. Since the corrupt data set consists only of random entries, then all orders associated with this data set are approximately equally likely- consequently, they are accepted at random, and the networks sampled from them do not represent any true, underlying relations between the variables. As a result, we do not expect any connection to the hyper network, which translates into low values for the hyper parameter  $\beta_4$ .

In figure 19 we illustrate trace plots of the sampled hyper parameters for all data sets. It appears that the method succeeds in identifying the corrupt data set: The hyper parameters associated with the true data sets have a wider range and reach up to higher values, in contrast to the hyper parameter corresponding to the fourth data set, which is systematically close to zero.

Finally, we want to examine whether introducing node ordering into the original model of Werhli and Husmeier improves convergence and mixing of the Markov chain. Since structure MCMC moves have proven unreliable in terms of convergence, as detailed in ([24]), we want to investigate whether order MCMC moves can avert, or, at least, improve, this problem. To this end, we compare the two, slightly different models, on simulated data.

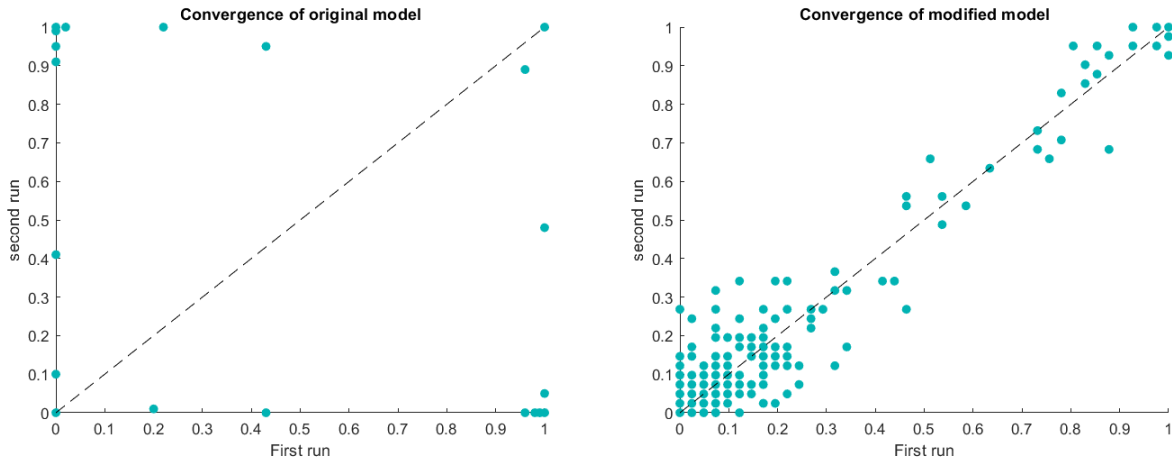


Figure 20: Convergence of the original(left panel) versus the modified (right panel) coupling scheme..

In figure 20, we see the comparison of edge score estimates, achieved by the original (left panel)

and the modified (right panel) version of the coupling scheme, over two independent simulations, on one of the true data sets. The simulations we run for  $3 \cdot 10^3$  steps. Whereas the modified version of the model seems to already have converged, the original model, which is based on structure MCMC moves is still far from convergence. This experiment indicates that introducing node ordering is meaningful, as it accelerates convergence and mixing of the Markov chain.

## 5.4 Psychometric data

After finalizing our strategy, we proceed on the analysis of the psychometric data set. As a first approach, we perform the simple cross validation scheme for the patients, using mix, static, and dynamic BN, as described in section 4.2. In figure 21, we illustrate the missclassification rate of all three models on the four patient groups.

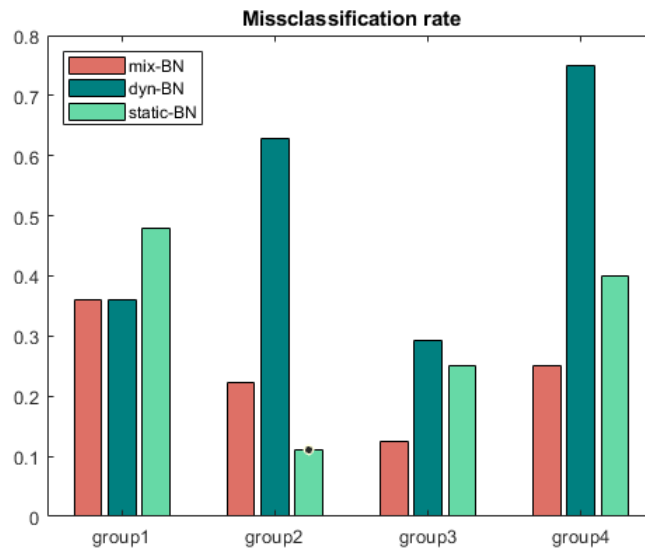


Figure 21: Missclassification rates on patients from mix, dynamic and static BN

It is clear that mix BN achieves higher classification accuracy than the two pure models on average. Surprisingly enough, dynamic relations between variables perform rather poorly as predictors- especially in the extreme case of the fourth data set, where missclassification rate of the dynamic models climbs up to 76%. The mix BN model, which has more information in disposal, coming from both contemporaneous and the static interactions between the variables, outperforms the pure models, with the exception of the second group, where the static model achieves a better classification accuracy.

We then applied the node-ordering version of the Werhli Husmeier model on the psychometric data set. Although this scheme gave very promising results on synthetic data, we did encounter significant convergence issues- in fact, convergence was only achieved for the third patient group. The scatter plots in figure 23 illustrate the different estimates, obtained from two independent runs of the model on the psychometric data. The deviation from the diagonal indicates a lack of convergence for all patient groups, except for patient group 3. Recall that in this model, a prior distribution over possible parent sets for every model is imposed. The prior terms, which are upper bounded by 1, result in a shrinkage of the posterior of a parent set, when it deviates from the corresponding parent set in the hypernetwork. Therefore, on an intuitive level, acceptance of MCMC moves is hindered, as it now depends on two factors: the likelihood term and the prior term. This results in a significant delay of the mixing of the Markov chain.

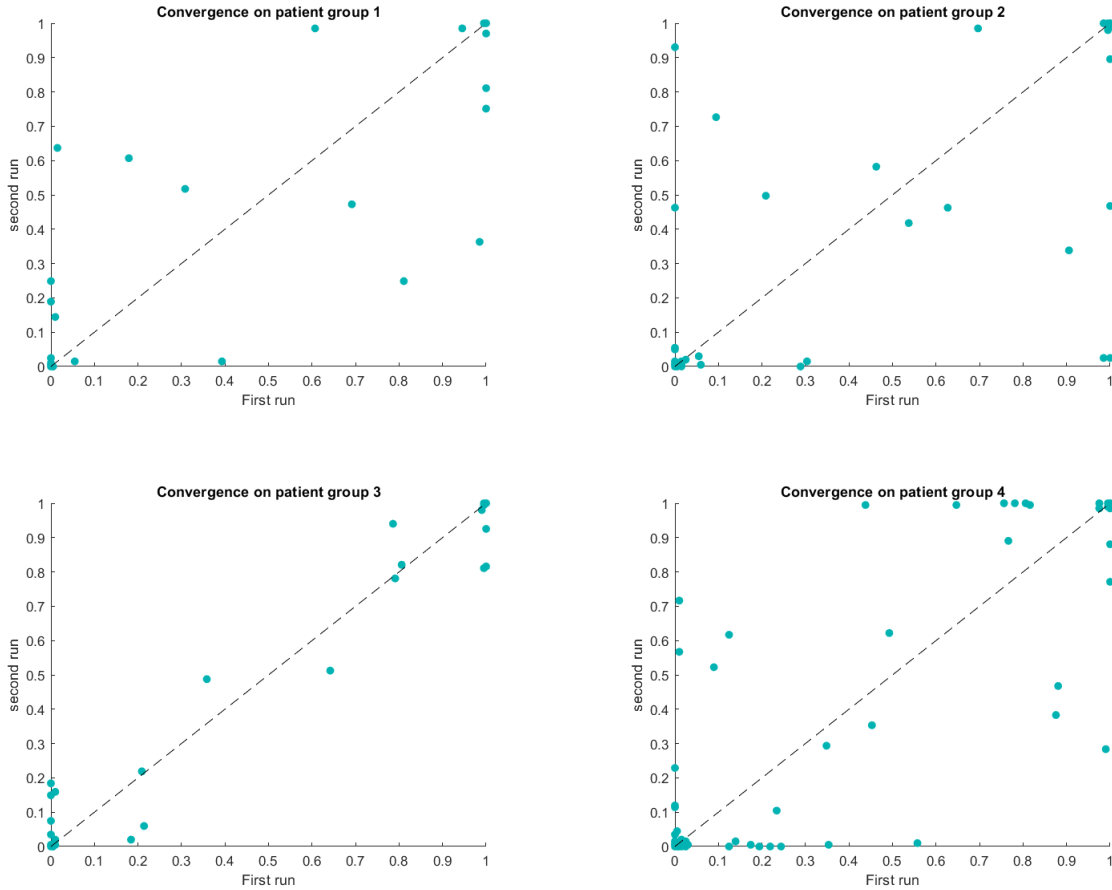


Figure 23: Scatter plots for estimates of the edges, for the four patient groups.

In figure 25 we illustrate the four networks that were inferred from the four data sets. These results are not, however, representative and should be interpreted with caution, since there was

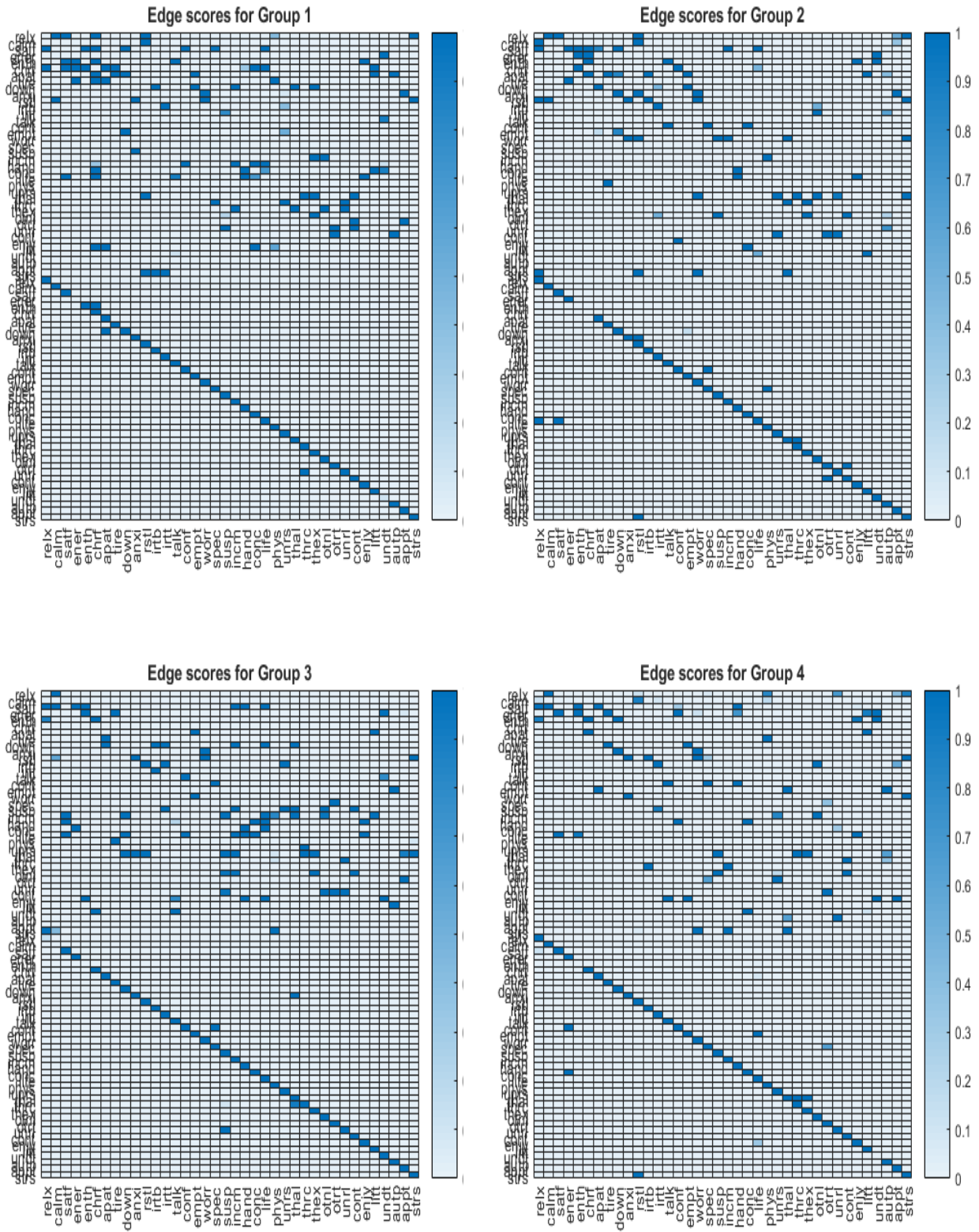


Figure 25: Heatmaps of the inferred symptom networks, for the four patient groups.

disagreement between independent simulations on a small, yet significant fraction of the inferred edges for patient groups one, two and four.

Although convergence was not achieved, there was stronger agreement between independent simulations on the values of the hyperparameters  $\beta$ . In figure 27, we illustrate trace plots of the  $\beta$  values accross the simulation, for all four groups. It appears that the network associated with the first data set is the only one that deviated significantly from the hyper network, whereas the other three networks coupled more strongly with the hyper network and thus, with one another. As patients of the first group are the ones showing the milder symptoms, it is probable that they share less common symptoms with patients from the other groups, who suffer from stronger symptoms.

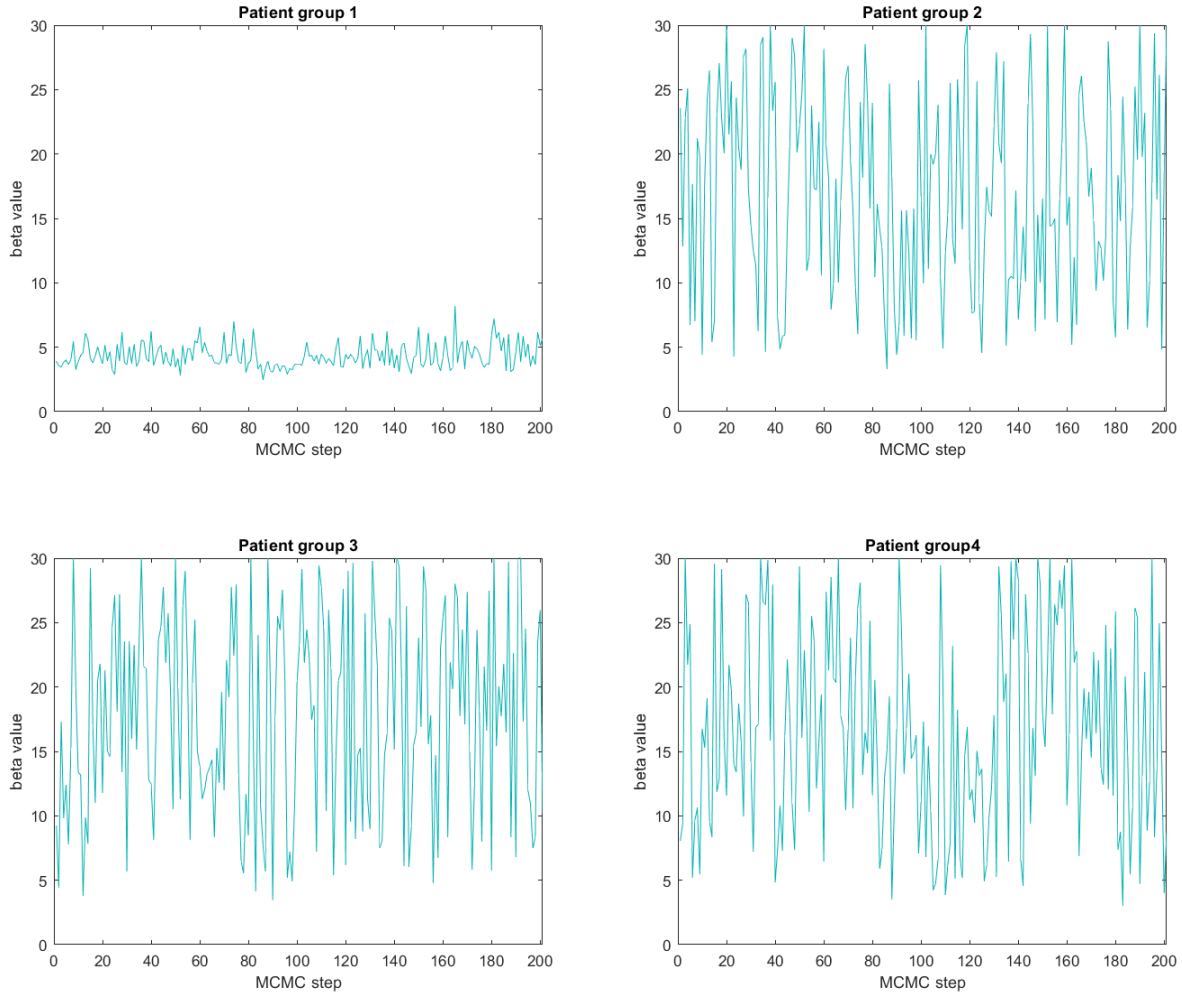


Figure 27: Traceplots of the hyperparameters beta. Larger  $\beta$  values indicate a stonger coupling with the hypernetwork.

Our approaches on the psychometric data sets were very computationally expensive. For the cross

validation scheme on the patients, we needed to run new order-MCMC simulations for every patient. The reason is that the group specific data set changes for every patient, as we extract the part of the data which corresponds to the specific patient. The running time of the algorithm in real life was approximately three days. The coupling scheme was also very costly. The model run for approximately 6 days on the Peregrine cluster, and even then, convergence was not achieved. .

## 6 Discussion-conclusion

The goal of this thesis was to introduce a new mix Bayesian network model, that combines features of static and dynamic Bayesian networks, and is able to infer static and dynamic interactions simultaneously from temporal data.

We firstly recapitulated some previous work on static and dynamic Bayesian networks (2.1.1. and 2.1.2) Then, in subsection 2.2, we introduced this new model. We presented a way to represent its connectivity structure by a block matrix, and also discussed a modified version of Chickering’s algorithm, which allows us to extract the CPDAG of a mixed Bayesian network.

Then, we discussed the BGe scoring metric of Geiger and Heckerman (2.3), which allows us, under some fairly weak assumptions, to derive a closed form expression for the marginal likelihood. In the first two subsections (2.3.1 and 2.3.2), we showed how the BGe score function of a static and a dynamic Bayesian network can be derived. In the last subsection (2.3.3), we presented how, by applying some straightforward modifications, we can also compute the BGe score of a mixed BN.

We proceeded by discussing a search strategy in order to learn network structures from data, under subsection (2.4.3). To this end, we presented two different MCMC approaches, structure MCMC (2.4.4.1), where the Markov chain is constructed in the space of structures, and order MCMC (2.4.4.2,) where the Markov chain is constructed in the space of orders. We discussed how order MCMC leads to a faster convergence on the Markov chain, relative to structure MCMC.

We concluded our methods section, with a coupling scheme that allows us to handle disjunct data sets which stem from similar, underlying networks. The model is proposed by Werhli and Husmeier ([24]), but we employ a modified version that uses node ordering (2.6.2), in order to improve convergence.

Afterwards, the methods were applied on synthetic data, generated from the RAF pathway 3.1 and on the psychometric data set 3.2. The results are outlined in section 5.

The simulation study indicated a very good performance of the new model, as it achieved very high scores on all settings. Comparisons between models showed that the new model competes

with the pure models on "pure" data. Especially on pure temporal data, the dynamic and the mixed Bayesian network models behaved very similarly.

Results on psychometric data showed that there was important information that could not be seen by the Dynamic Bayesian network model. The underlying contemporaneous interactions between symptoms were very strong, and therefore the dynamic model provided inadequate insights on the domain. Especially in the cross validation scheme, the dynamic Bayesian model achieved very low classification accuracy, which indicates that the dynamic relations between the variables are not good predictors for distinguishing between groups. In this experiments, the mixed Bayesian network model had a very satisfactory performance, as it outperformed both models.

We can conclude that in case a domain consists of temporal data with both dynamic and static underlying interactions, the mixed model is the best option between the three models. The pure models are able to only detect the dominant corresponding features, and our experiments indicated that they tend to infer a significant number of wrong edges. Moreover, the coexistence of dynamic and static edges provides additional insights that cannot be obtained by applying the pure models separately, as in the mixed models, some of the symmetries of the static edges can break because of the presence of the dynamic edges. In other words, static edges that are bidirectional from a static point of view can be rendered compelled by the dynamic edges, which lifts the ambiguity and allows us to make for specific statements about the interactions.

However, the mixed model comes with the shortcoming that it doubles the variables of the domain, which leads to a big computational overhead, and also hinders convergence, especially when combined with structure MCMC simulations. In these simulations, our experiments indicated that the mixed model does not perform that well when we have small data sets (low  $m$ ) and as the number of nodes increases. We proposed order MCMC as an alternative, in order to improve convergence, however this method introduces bias into the model. In the past, there have been different attempts to improve convergence in structure MCMC, without having to pay the price of bias, such as the REV move [15] and Partition MCMC [32]. However, these techniques have only been introduced for the static Bayesian network model, and it is not trivial to adapt them into the mixed setting, because of the presence of the dynamic edges.

## References

- [1] Balian Roger. *From microphysics to macrophysics : methods and applications of statistical physics / Roger Balian ; translated by D. ter Haar and J. F. Gregg. Volume I.* Texts and monographs in physics. Springer-Verlag, Berlin New York Heidelberg [etc, cop. 1991.
- [2] Dan Geiger and David Heckerman. Learning gaussian networks. In *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*, UAI'94, page 235–243, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
- [3] Jack Kuipers, Giusi Moffa, and David Heckerman. Addendum on the scoring of gaussian directed acyclic graphical models. *The Annals of Statistics*, 42(4):1689–1691, 2014.
- [4] Marco Grzegorzczuk, 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, 07 2008.
- [5] Sanne H. Booij, Marieke Wichers, Peter de Jonge, Sjoerd Sytema, Jim van Os, Lex Wunderink, and Johanna T. W. Wigman. Study protocol for a prospective cohort study examining the predictive potential of dynamic symptom networks for the onset and progression of psychosis: the mapping individual routes of risk and resilience (mirorr) study. *BMJ Open*, 8(1), January 2018.
- [6] Denny Borsboom and Angélique O.j. Cramer. Network analysis: An integrative approach to the structure of psychopathology. *Annual Review of Clinical Psychology*, 9(1):91–121, March 2013.
- [7] Wray Buntine. A guide to the literature on learning probabilistic networks from data. *Knowledge and Data Engineering, IEEE Transactions on*, 8:195 – 210, 05 1996.
- [8] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*, volume 81. 01 1993.
- [9] Eunice Yuh-Jie Chen, Arthur Choi Choi, and Adnan Darwiche. Enumerating equivalence classes of bayesian networks using ec graphs. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 591–599, Cadiz, Spain, 09–11 May 2016. PMLR.

- [10] Max Chickering. Learning equivalence classes of bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, February 2002.
- [11] S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano. Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. In *Proceedings of the 2003 IEEE Bioinformatics Conference, CSB 2003*, Proceedings of the 2003 IEEE Bioinformatics Conference, CSB 2003, pages 104–113, United States, 2003. Institute of Electrical and Electronics Engineers Inc. 2nd International IEEE Computer Society Computational Systems Bioinformatics Conference, CSB 2003 ; Conference date: 11-08-2003 Through 14-08-2003.
- [12] Frank Dondelinger, Sophie Lèbre, and Dirk Husmeier. Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Machine Learning*, 90(2):191 – 230, February 2013.
- [13] Nir Friedman and Daphne Koller. Being bayesian about network structure, 2013.
- [14] Marco Grzegorzcyk. An introduction to gaussian bayesian networks. In *Systems Biology in Drug Discovery and Development*, pages 121–147. Springer, 2010.
- [15] Marco Grzegorzcyk and Dirk Husmeier. Improving the structure mcmc sampler for bayesian networks by introducing a new edge reversal move. *Machine Learning*, 71(2-3):265–305, June 2008.
- [16] David Heckerman. A tutorial on learning with bayesian networks, 2020.
- [17] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning bayesian networks: The combination of knowledge and statistical data, 2015.
- [18] Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282, 11 2003.
- [19] David Heckerman and Dan Geiger. Learning bayesian networks: A unification for discrete and gaussian domains. pages 274–284, 01 1995.
- [20] Radford M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, 1993.

- [21] Mahdi Shafiee Kamalabad, Alexander Martin Heberle, Kathrin Thedieck, and Marco Grzegorzcyk. Partially non-homogeneous dynamic Bayesian networks based on Bayesian regression models with partitioned design matrices. *Bioinformatics*, 35(12):2108–2117, 11 2018.
- [22] Karen Sachs, Omar Perez, Dana Pe’er, Douglas Lauffenburger, and Garry Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science (New York, N.Y.)*, 308:523–9, 05 2005.
- [23] David A Spade. Markov chain monte carlo methods: Theory and practice. *Principles and Methods for Data Science*, 43:1, 2020.
- [24] ADRIANO V. WERHLI and DIRK HUSMEIER. Gene regulatory network reconstruction by bayesian integration of prior knowledge and/or different experimental conditions. *Journal of Bioinformatics and Computational Biology*, 06(03):543–572, 2008.
- [25] David Balding, Chris Cannings, and Martin Bishop. Handbook of statistical genetics. 08 2007.
- [26] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- [27] Dan Geiger and David Heckerman. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *The Annals of Statistics*, 30(5):1412 – 1440, 2002.
- [28] David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232, 1995.
- [29] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI ’90, page 255–270, USA, 1990. Elsevier Science Inc.
- [30] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- [31] David Maxwell Chickering. A transformational characterization of equivalent bayesian network structures. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI’95, page 87–98, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

- [32] Jack Kuipers and Giusi Moffa. Partition mcmc for inference on acyclic digraphs. *Journal of the American Statistical Association*, 112(517):282–299, Jan 2017.
- [33] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9(4):309–347, October 1992.

# Appendix

We will now provide a brief description of the software developed in this thesis. The programming tool that we employ is MATLAB. The code has been provided for my supervisors, and is available upon request. We first describe the functions we created for structure MCMC simulations. Afterwards, we move on to our order MCMC simulation. Subsequently, we describe the function we created for the modified version of the Werhli and Husmeier model (coupling scheme). We conclude with some additional functions that we created for our analysis. An illustration of the code for a structure MCMC simulation on mixture data is provided in the end of the appendix.

## Structure-MCMC

In our implementation of *Metropolis Hastings structure-MCMC*, we created three functions, corresponding to the static, dynamic and mix models. Below we give a general description of some of the main inner functions we created. Most functions have a static, dynamic and mixed counterpart and are called when we employ the corresponding model. The static model can be employed if we call our `MCMC_static` function, whereas for the dynamic case, we created the `dyn_MCMC` function. For our mixed Bayesian network model, we created the `MCMC_mix` function. These functions can be called with the following commands:

```
1
2 %mixed Bayesian networks
3 [sampled_nets_mix , BGe_scores_mix] = MCMC_mix( G_init_stat ,
         G_init_dyn , iter , D, T0, mu_0 , v , a , fan_in , thin_steps , burn_steps ,
         self_loops );
4
5 %static Bayesian networks
6 [sampled_nets_stat , BGe_scores_stat] = MCMC_static( G_init , iter , D,
         T0_stat , mu_0_stat , v , a , fan_in , thin_steps , burn_steps );
7
8 %dynamic Bayesian networks
9 [sampled_nets_dyn , BGe_scores_dyn] = dyn_MCMC( G_init , iter , D, T0_dyn ,
         mu_0_dyn , v , a , fan_in , thin_steps , burn_steps , self_loops );
```

The functions return a collection of high scoring networks (`sampled_nets`), and their corresponding BGe scores (`BGe_scores`). The input arguments `G_init` in `MCMC_static` and

`dyn_MCMC` correspond to an initial network/state. In the `MCMC_mix` function, we have the input arguments, `G_init_stat` and `G_init_dyn`, which correspond to an initialization for the static and dynamic part of the initial mixed network state, respectively. In all three functions, the values of the hyper parameters, `T_0`, `mu_0`, `a`, `v`, and a positive integer corresponding to the fan-in restriction (`fan_in`) are required as input. In addition, the number of MCMC steps needs to be specified (`iter`), as well as the length of the burn-in phase (`burn_steps`) and the frequency of sampling (`thin_steps`). When it comes to the dynamic or the mix models, an extra Boolean argument is included (`self_loops`), which tells the function whether self loops are accepted as valid edges.

Within the main loop, a new network is proposed from the neighbourhood of the current network, with the `pick_neighbour` function. As an input argument to this function, we plug three matrix arguments, namely `addition_matrix`, `removal_matrix`, `reversal_matrix` (in static and mix case). These are binary matrices, that represent the valid edge operations: additions, removals, and reversals. The matrices are used not only for the determination of a valid neighbour of the current network, but also for the computation of the proposal probabilities-note that the sums of these matrices are the number of neighbours of the current network.

Then, the acceptance probability of the newly proposed state must be computed. To this end, the BGe score, must be calculated. In order to avoid underflow-overflow issues, we work on logarithmic scale-that is, we consider the logarithms of the BGe scores. The BGe score was detailed in subsection 2.3. Assuming that the domain consists of  $N$  variables and  $m$  observations, the equation for the complete network for the static case is given in 16, whereas the BGe score is given in 14, as a factorization of *local scores*. For the sake of efficiency, the calculation of the local score, can be simplified. As explained in the supplementary material of [3], a single term,  $\Psi(D_i^{\pi_i})$ , in the product of the BGe score in 14 (for the static case) is given by:

$$\Psi(D_i^{\pi_i}) = \pi^{-\frac{m}{2}} \left( \frac{v}{v+m} \right)^{\frac{1}{2}} \frac{\Gamma(\frac{a+m-N+p+1}{2})}{\Gamma(\frac{a-N+p+1}{2})} \frac{\det(T_0^{\{\pi_i\}})^{\frac{\alpha-N+p+1}{2}}}{\det(T_0^{\{\pi_i \cup X_i\}})^{\frac{\alpha-N+p}{2}}} \frac{\det(T_{D,m}^{\{\pi_i\}})^{\frac{\alpha+m-N+p}{2}}}{\det(T_{D,m}^{\{\pi_i \cup X_i\}})^{\frac{\alpha+m-N+p+1}{2}}} \quad (40)$$

where  $\det(\cdot)$  denotes the determinant of its input matrix argument, whereas  $\Gamma(\cdot)$  denotes the Gamma function. The parent set for node  $X_i$  is denoted as  $\pi_i$  and its cardinality is denoted as  $|\pi_i| = p$ . The matrix  $T_{D,m}$  is given in 12 and the matrices  $T_0^{\{S\}}$  and  $T_{D,m}^{\{S\}}$  are the submatrices of  $T_0$  and  $T_{D,m}$ , obtained if all rows corresponding to variables not in  $S$  are discarded. This equation stems from the observation that if we consider the local score, instead of handling the numerator and the denominator in the local score independently, many of the terms cross out. Equation 40

gives the local BGe score of a static network, however, since for the dynamic and the mixed cases, the same computational steps are followed, the equation applies to these cases as well, after some straightforward modifications.

For the efficient calculation of the BGe score, we can pre compute the constant prefactors. When it comes to the derivation of the determinants, we can reduce computational costs by computing  $\det(T_{D,m}^{\{S\}})$  using Cholesky decomposition instead of directly calling the `det()` MATLAB command, which uses LU decomposition, keeping in mind that  $T_{D,m}$  is a real, positive definite, symmetric matrix. Furthermore, in case where we set the hyper parameters to be uninformative, we can compute  $\det(T_0^{\{S\}})$  more efficiently as well. In case the hyperparameters are "suppressed",  $T_0$  is of the form  $cI_N$ , where  $c$  is a scalar and  $I_N$  is the  $N$  dimensional identity matrix. In this scenario, we have  $\det(T_0^{\{S\}}) = c^{|S|}$ , where  $|S|$  is the cardinality of  $S$ .

Recall that, in structure MCMC, the new network/state stems from a local modification in the structure of the current network/state, namely, an edge addition, removal, or reversal. Observe that if an edge  $X_i \rightarrow X_j$  is either added or removed, then the only parent set that is affected, is the parent set of node  $X_j$ . In terms of local scores, this means that the local scores of all nodes, except  $X_j$ , do not change after an edge addition or removal. In case edge  $X_i \rightarrow X_j$  is reversed into  $X_i \leftarrow X_j$ , then only the parent sets of  $X_i$  and  $X_j$  change. As a consequence, if we consider the computation of the acceptance probability in 24, after a valid edge operation between nodes  $X_i$  and  $X_j$  then the terms of the BGe scores corresponding to the rest of the nodes cross out, which reduces the acceptance probability to a ratio of the local score of  $X_j$  (and  $X_j$  and  $X_i$  is case of an edge reversal) times the Hastings ratio, that is, the ratio of the proposal probabilities. The logarithm of the score of the newly proposed network can then be computed from the logarithm of the score of the current network, simply by subtracting to logarithm of the affected local score and adding the logarithm of the scores after the valid edge operation.

The derivation of the *Hastings ratio*, is a rather costly factor within an MCMC step, as it requires the enumeration of all possible neighbours of the newly proposed network. In our most demanding settings (psychometric data set) we can, the sake of efficiency, consider omitting its calculation by assuming that it is always equal to one.

If the new network is accepted, then we need to update the current network/ state, and its neighbourhoods, namely, the matrices representing the reachable states- `addition_matrix`, `removal_matrix`,

`reversal_matrix` (in static and mix case case). For this purpose, we created a function called `update_matrices`, whose role is to update the neighbourhood of a network efficiently, and to remove the neighbours that violate any restrictions imposed by the models (self loops, fan-in). For instance, if node  $i$  has  $k$  incoming edges, and  $\text{fan\_in} = k$ , then no edge pointing towards  $i$  can be added in the next step.

## Order-MCMC

For the implementation of *order MCMC* on mixture data, we created a function called `order_MCMC`. This function returns a collection of high scoring orders (`trajectory_orders`), the corresponding scores (`scores`), a collection of corresponding networks (`sampled_networks`), sampled from the orders, as described in subsection 2.4.4.2, and a cell array, `extracted_fams`, in which the families that are consistent with each order in the sample are stored. As input arguments, we need the node specific data matrices, stored in a 3D array  $D$ , the corresponding  $T_m$  matrices (3D array), an initial order/state, the hyperparameters  $T_0, \mu_0, v, a$ , the fan in restriction  $k$ , a positive integer  $C$  specifying the number of highest scoring parents that will be considered for every node. Furthermore, the number of MCMC steps (`iter`), the length of the burn phase (`burn_steps`), and the frequency of sampling (`thin_steps`) must be specified. The `order_MCMC` function can be called with the following command:

```
1 %orderMCMC simulation
2 [ trajectory_orders , scores , extracted_fams , sampled_networks ] =
   order_MCMC( order , iter , burn_steps , thin_steps , D , k , C , a , T0 , Tm , v ,
   self_loops );
```

Some pre-computations are performed before the MCMC simulation starts: For each node, a collection of the highest scoring candidate families are computed with the `pre_compute_families` function. Within this function, the  $C$  highest scoring parents (using a `pre_compute_edges` function) of each nodes are identified, and are then combined into promising parent sets for the node. Then the top-scoring families for the nodes are stored in a cell array, `top_fams`, whose  $i$ -th cell contains the parent sets for node  $i$ , with the corresponding scores, in descending order.

An additional inner function is created, called `extract_families`, whose role is to return the families that are consistent with the given order, stored in a cell array, the corresponding scores, and a network, that is sampled from the order. This is required for the computation of the prob-

ability of the data given an order, as can be seen in 25. This function runs over all high scoring families for node  $i$ , stored in the `top_fams` cell array, and extracts only those that do not contain any nodes that succeed  $i$  in the given order. This process is followed for all nodes  $i$ , starting from the last node in the order.

This function is developed so that it can be employed on an order that is proposed by node flipping, within an MCMC step. The indices of the nodes that exchanged places in the order have to be plugged in as input. Then, the function only runs on the "suborder", consisting only of the flipped orders and those nodes that stand between them in the order, as the consistent families for the rest of the nodes remain unchanged. With this in mind, the order of the new score can be computed by subtracting the score of the suborder before the flipping and adding the score of the sub order after the flipping. After the score of the newly proposed order is computed with the `extract_families` function, it is used to compute the acceptance probability of the order.

## The coupling scheme

For the implementation of the modified model of Werhli and Husmeier, we created a function called `wh_order`, which can be called with the following command:

```
1 %modified werhli-husmeier model
2 [ trajectory_order , scores , traj_networks , extracted_fams , betas ,
   hyper_net ] = wh_order ( hyper_net , orders_all , b , iter , burn_steps ,
   thin_steps , D_cube , k , C , a , T0 , Tm_cube , v , self_loops , fan_in );
```

Assume we have  $I$  data sets and  $N$  variables in the domain. As input arguments, we need to specify the initial states: An initialization for the hyper network (`hyper_net`),  $I$  initial orders associated with each data set, stored in an  $I$  by  $N$  matrix `orders_all`, and an  $I$  dimensional vector of initial hyper parameters  $\beta$  (`b`). Moreover, the hyper parameters  $T_0, \mu_0, a, v$ , as well as the Boolean argument `self_loop` have to be specified. Since node-ordering is introduced, the parameters  $C$  and  $k$  also need to be determined. The `fan_in` argument concerns the hyper network and is only optional (we included no fan in restriction for the hyper network). The arguments `D_cube` and `Tm_cube` are both  $I$  dimensional cell arrays, where the node specific data and  $T_{D,m}$  matrices are stored, for all  $I$  data sets. Finally, the arguments `iter`, `burn_steps`, `thin_steps` specify the length of the simulation, the burn in and the thinning phases. The function returns 6 arguments: A 3D matrix (`trajectory_order`), where the sampled orders associated with each data set are

stored, and the corresponding scores, stored in a matrix `scores`. The `I` dimensional cell array `traj_networks` contains a sample of networks for each data set, stored in 3D matrices. The `I`-dimensional cell array `extracted_fams` contains the families that are consistent with the orders associated with each data set, stored in sub cells. The matrix `betas` contains the trajectory of the hyperparameters  $\beta$ , for the data sets. Finally, the output argument `hyper_net` corresponds to the hyper network.

The function follows the sampling process as detailed in subsection 2.6.2. Within this function, two cell arrays are defined, called `prior_scores` and `prior_terms`, in which the likelihoods and the priors of all consistent parent sets given an order are stored. In the first sampling step, a new order is proposed for all data sets. To this end, a function called `propose_new_order` was created. Every time a new order is accepted, the parent sets that are consistent with the order change. The parent sets that are consistent with the new order are extracted with the same process as in the `extract_families` function, described in the previous subsection. If the new order is accepted, the arrays `prior_scores` and `prior_terms` need to be updated.

Subsequently, new hyper parameters  $\beta_i$  are proposed, for all data sets. Every time a new hyper parameter is accepted, then the `prior_terms` cell array is updated. In the final sampling step, a new hyper network is then proposed with the function `propose_hypernetwork_wh`. This new hyper network is proposed with an edge operation, with the help of a function similar to `pick_neighbour` function, which is called in the structure-MCMC simulations. Every time a hyper network is accepted, the cell array `prior_terms` need to be updated.

Updating the prior terms implies that the normalization constant has to be computed. This is a very costly calculation, so we came up with a formula for its efficient computation. We denote  $|U|$  the cardinality of the set  $U$ , and  $\pi_n$  the set of all possible parent sets for node  $n$ . We define the family of sets associated with a node  $n$ , that comply with the fan in restriction  $k$ , and have  $l$  miss matches with the corresponding parent set in the hypernetwork, denoted  $U^*$ , as  $U_l = \{U \in \pi_n : |U \oplus U^*| = l, |U| \leq k\}$ , where  $\oplus$  is the symmetric difference between two sets. Denoting  $p^* = |U^*|$ , we can rewrite the single node partition function as:

$$\begin{aligned} Z_n(\beta, \mathcal{M}^*) &= \sum_{U \in \pi_n} e^{-\beta|U \oplus U^*|} \\ &= \sum_{l=0}^c |U_l| e^{-\beta l} \end{aligned}$$

with  $c = \min\{N - 1, p^* + k\}$ . The sum runs over all possible numbers of mismatches between a parent set for  $n$  and the corresponding parent set of  $n$  in the hyper network. The problem then reduces to the enumeration of all possible sets  $U \in U_l$ , which is a rather simple combinatorial problem. The key insight is that every set  $U \in U_l$  has the property of having  $l$  mismatches with the set  $U^*$ :

$$\begin{aligned} U \oplus U^* = l &\Rightarrow |U| + |U^*| - 2|U \cap U^*| = l \Rightarrow \\ &= |U \cap U^*| = \frac{p^* + p - l}{2} \end{aligned}$$

where  $p = |U|$ . Therefore, a set  $U$  with cardinality  $p$  can be constructed by choosing  $\frac{p^* + p - l}{2}$  elements from  $U^*$  (the ones they have in common), and then filling the rest of the  $p - \frac{p^* + p - l}{2}$  spaces in  $U$  (the elements they disagree on) with elements not in  $U^*$ , which leaves  $N - p^* - 1$  choices, considering that  $n$  is also excluded from the possible choices.

$$|U_l| = \sum_{p=c_{min}}^{c_{max}} \binom{p^*}{\frac{p^* + p - l}{2}} \binom{N - p^* - 1}{\frac{p^* - p + l}{2}}$$

with  $c_{min} = \max\{0, p^* - l\}$  and  $c_{max} = \min\{k, p^* + l\}$ , and  $k$  is the fan in restriction. For the computation of the normalization constant, we created a function called `compute_Z`, whose role is, given the hyperparameter  $\beta$ , the fan-in restriction and the hyper network as input arguments, to return the number

$$Z(\beta, \mathcal{M}^*) = \prod_n Z_n(\beta, \mathcal{M}^*)$$

## Additional helper fuctions

### Extraction of CPDAG and mixed CPDAG

For the extraction of the CPDAG for the static case, we implemented Chickering's algorithm [10], by creating a function called `CPDAG_from_DAG`. To this end, two inner functions were created: a function called `topological_order`, which returns the order of its input DAG argument, and the `order_edges` function, which returns an ordering of the edges of its input DAG argument. Then, the `CPDAG_from_DAG` function classifies the edges as either compelled or reversible, and thus extracts the CPDAG of its input DAG argument. The `CPDAG_from_DAG` function is utilized for the extraction of the mixed- CPDAG of a mixed network, which is achieved by our `CPDAG_from_DAG_mix` function. The algorithm is detailed in subsection 2.2: This function re-

quires a valid mixed network as an argument. The static part of its incidence matrix is extracted, and then extended by two rows and to columns, which correspond to the temporarily added pseudoparents. Then, the `CPDAG_from_DAG` is called on this extended matrix.

## Data generation

Our simulation study was performed on synthetic data, as described in subsection 3.1. To this end, we created three functions that can create static, dynamic and mixture data from the RAF signalling pathway.

```

1  %generate mixture data from RAF signalling pathway
2  [D_mix , G_stat , G_dyn , G_mix]=make_mixed_data_RAF(m) ;
3
4  %generate static data from RAF signalling pathway
5  [D_stat , G_static]=make_synthetic_data(m) ;
6
7  %generate dynamic data from RAF signalling pathway
8  [D_dyn , G_dynamic]= make_data_dyn_RAF(m) ;

```

The input argument `m` indicates the number of observations of our domain. Mixture data is generated by the `make_mixed_data_RAF` function, which returns an 11 by `m` data matrix, `D_mix`, a mixed network `G_mix`, together with its static and dynamic parts (`G_stat`, `G_dyn`). Pure data is generated by the `make_synthetic_data` (static) and `make_data_dyn_RAF` (dynamic) functions, which return 11 by `m` matrices of pure static `D_stat` and pure dynamic `D_dyn` data, together with the true networks `G_static` and `G_dynamic` respectively.

## Model evaluation

As a scoring metric for our models, we use the precision among the highest scoring edges (see subsection 2.5). To this end, a function called `precision` was created. This function requires the true graph, the sample of networks (or their CPDAGs, depending on the model that we employ) as well as a positive integer `x`, which indicates the number of highest scoring edges that will be extracted. Then, the function returns the fraction of these edges that are among the true edges.

The AUROC values can be computed using MATLAB's `perfcurve` function.

## Example

In this example, we illustrate how we can call the `MCMC_mix` function to perform a structure-MCMC simulation on mixture data:

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%EXAMPLE
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3
4  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% STRUCTURE MCMC ON MIXED DATA
5  iter=100000; burn_steps=50000; thin_steps=100; %determine number
   of MCMC steps , burn and thin phase
6  m=100; n=22;
7  self_loops=1; fan_in=n;
8
9  [D,~,~,G]=make_mixed_data_RAF(m); %generate static data from RAF
   signalling pathway (n=11)
10 G_t=CPDAG_from_DAG_mix(G);          % extract CPDAG of true
   network G
11 a=n+2; mu_0=zeros(1,n); v=1;          %initialize
   hyperparameters(uninformative prior)
12 T0=0.5*eye(n);
13
14 G_init=zeros(n/2,n/2);
15 [sampled_nets_mix , mix_BGE_scores]= MCMC_mix( G_init , G_init , iter ,
   D,T0,mu_0,v,a,fan_in , thin_steps , burn_steps , self_loops );
16
17 sample_size=length( mix_BGE_scores ); %determine number of
   networks in the sample
18
19 CPDAGS_mix=zeros(n,n/2,sample_size); %extract CPDAGs from all
   networks in the sample
20 for net=1:sample_size
21   CPDAGS_mix(:, :, net)=CPDAG_from_DAG_mix( sampled_nets_mix(:, :, net
   ));
22 end
23
```

```

24  x=20;                %evaluate model using precision function
25  score_mix=precision( G_t ,CPDAGS_mix , x );
26  %%%%%%%%%%%

```