The Minimal Hoppe-Beta Prior Distribution for Directed Acyclic Graphs and Structure Learning

Timo J.T. Koski*[1], John M. Noble†[2] and Felix L. Rios‡[1]

[1]Department of Mathematics, KTH Royal Institute of Technology
[2]Institute of Applied Mathematics and Mechanics, University of Warsaw

December 17, 2015

Abstract

The main contribution of this article is a new prior distribution over directed acyclic graphs intended for *structured* Bayesian networks, where the structure is given by an *ordered block model*. That is, the nodes of the graph are objects which fall into categories or *blocks*; the blocks have a natural ordering or ranking. The presence of a relationship between two objects is denoted by a directed edge, from the object of category of lower rank to the object of higher rank. The models considered here were introduced in Kemp et al. [8] for *relational data* and extended to multivariate data in Mansinghka et al. [13].

We consider the situation where the nodes of the graph represent random variables, whose joint probability distribution factorises along the DAG. We use a minimal layering of the DAG to express the prior. We describe Monte Carlo schemes, with a similar generative that was used for prior, for finding the optimal a posteriori structure given a data matrix and compare the performance with Mansinghka et al. and also with the uniform prior.

1 Introduction

1.1 Background: The Ordered Block Model and Bayesian Networks

The *ordered block model* was introduced by Kemp et al. [8]. The ordered block model is an interesting and versatile idea for the situation where classification of individuals is to be inferred from observing how the individuals relate to each other (relational data). In the ordered block model there is a hierarchy, i.e., a distinct ordering or ranking, of a number of classes of a set of variables. There is assumed to exist a compatible directed acyclic graph (DAG) for the variables.

*tjtkoski@kth.se
†noble@mimuw.edu.pl
‡flrios@kth.se

Instances of the architecture of ordered block models have appeared independently, see e.g., the conceptual root cause analysis in [18] or in the causal independence model [6]. We consider modelling of situations, where there is a natural, but unknown, partition of the variables into ordered classes. We assume that the probability distribution over the variables may be expressed as a Bayesian Network, and express the DAG by its minimal layering, which should then capture the hierarchies in the problem domain after a learning procedure. The minimal layering is a DAG where there is only the possibility of a directed edge from node x to node y if and only if x belongs to a class of strictly lower rank than class of y.

Our minimal Hoppe-Beta prior probability is based on the notion of a unique minimal layering of the nodes in DAG, c.f., Tamassia [17], Healy and Nikolov [4], which exists for any DAG. The graph structure of the minimal layering implies the class assignments of the nodes in the DAG. If one wishes to infer class structure from a DAG, then the minimal layering represents as much of the class structure that can be inferred from data alone. There are alternative possible choices of class structure other than minimal layering. The point of the minimal layering is that data influences the class structure only through the update on the DAG structure.

One main advantage of the minimal Hoppe-Beta Prior, in comparison with Kemp et al. [8], is that there is a simple computable closed form expression for it, as established in this paper.

The posterior probability for a DAG reflects the degree in which the corresponding minimal layering fits data through the likelihood and the prior probability of the minimal layering.

1.2 Classification of Variables and Bayesian Networks

Data comes in the generic form of an $n \times d$ data matrix $\mathbf{x}$, where each row represents an independent instantiation of a discrete random d-vector with the joint probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$. Throughout, $\mathbf{x}$ will denote the matrix of instantiations and $\mathbf{X}$ the underlying random matrix. Also, we use $X = (X_1, \ldots, X_d)$ to denote the random vector (taken as a row vector, in accordance with the way that data is presented in a data matrix).

A collection of random variables $\{X_1, \ldots, X_d\}$ can be ordered in $d!$ ways and each gives rise to a factorisation: for a permutation σ of $\{1, \ldots, d\}$,

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|\mathrm{Pa}(\sigma(j))} \tag{1}$$

where, with the ordering σ, for each j, $\mathrm{Pa}(X_{\sigma(j)}) \subseteq \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\} =: \mathcal{P}_j^\sigma$ and $\mathcal{P}_j^\sigma$ denotes the σ-*predecessors* of $X_{\sigma(j)}$. $\mathbb{P}_{X_{\sigma(j)}|\mathrm{Pa}(\sigma(j))}$ is the conditional probability of $X_{\sigma(j)}$ given the parents $Pa(\sigma(j))$. For the ordering σ, for each j, the parent set $\mathrm{Pa}(\sigma(j))$ is taken as the smallest subset of $\mathcal{P}_j^\sigma$ such that Equation (1) is true. A factorisation may be expressed by a DAG, which has a directed arrow $j \to k$ if and only if $X_j \in \mathrm{Pa}(k)$. Such a factorisation, together with the corresponding DAG, is known as a Bayesian Network (or BN). A DAG is expressed as $G = (V, D)$, where V is the set of nodes and D is the set of edges $\subseteq E \times E$, and can also referred to as the DAG structure. The nodes can be identified with the variables $X_1, \ldots, X_d$.

We consider *classification of the variables* $X_1, \ldots, X_d$, where the number of classes is a priori unknown, as introduced in Mansinghka et al. [13], who extend the work of Kemp et al..

In many 'real world' problems, variables may be grouped into classes which indicate the nature of their probabilistic relations. For example, in the QMR-DT network, introduced by Shwe et al. [16], variables fall into two classes; *diseases* and *symptoms*, where various diseases may cause various symptoms. In Mansinghka et al. [13], the HEPAR II network, due to Oniśko et al. [14] is considered.

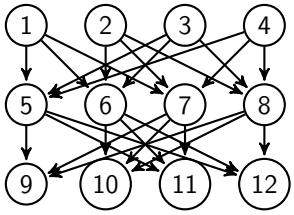

Figure 1: The HEPAR II network.

The DAG is shown in Figure 1. It is a Bayesian Network for analysing liver diseases where the variables (identified as nodes) can be divided into three classes; *risk factors*, *diseases*, *symptoms*. The conditional probabilities for the HEPAR II network have been learned from a database of medical cases.

In the Figure 1 the classes are $C_1 = \{1, 2, 3, 4\}$, $C_2 = \{5, 6, 7, 8\}$ and $C_3 = \{9, 10, 11, 12\}$. There are no edges (arrows in the figure) between the nodes belonging to the same class. If the classes are ordered as $C_1 < C_2 < C_3$, then all edges are seen to be from a lower rank class to a higher rank class. This is an example of an ordered block model.

The aim of a data analysis with an ordered blocks is therefore two-fold: to find a suitable classification of the variables and also *structure learning* [9]; namely, to find a DAG which encodes the direct influences that the variables have on each other, where the layers of the graph correspond to the classes of an ordered block model.

1.3 Block Model: Prior, Posterior and Structure Learning

The learning of the structure of the minimal layering from observed information means finding, under the constraint on the edges above, whether there is a directed edge, or there is no edge, or else no interaction was observed and the edge status is unknown. The choice of prior distribution so that it corresponds the prior beliefs about classes of the variables and ranking of classes, is clearly paramount for Bayesian algorithms for learning the DAG structure of an ordered block model.

Kemp et al. [8] introduce a prior, which we refer to as the *Hoppe-Beta Prior*, which is a joint distribution over classifications and graphs. In our case, we obtain a partition of the variables by the minimal layering, to be defined in section 2, of the DAG of the Bayesian Network.

There are various general approaches to the construction of informative prior distributions and corresponding learning of DAG structures, a concise survey is included in Angelopoulos and Cussens [1] (which does not mentioned Mansinghka et al. [13]). Kuipers and Moffa [10] show how to sample a DAG from a uniform distribution. The report [3] summarizes and analyses several techniques for

finding random DAGs.

The work of [15] invoked the "outpoints", when counting the number of labelled DAGs. The paper [11, section 4.1] describes these outpoints in detail. The layers of the outpoints are applied for constructing priors in Kuipers and Moffa [11], but there is a set of restrictions not present in our work.

Kemp et al. invoke a *Hoppe urn scheme* to construct a prior distribution over the class structure. This algorithm can be divided into two main steps: In the first step the nodes are partitioned into numbered classes using the Hoppe-Ewens urn scheme Hoppe [7](1984). The prior over classes is therefore constructed first, without reference to the prior over DAGs. The second step is the prior over DAGs conditioned on the classification. Firstly, the edge probability from a class a node to a class b node is generated using a suitable Beta distribution, these are independent for different pairs of classes and only edges from lower rank to higher rank classes are permitted. A DAG is then generated using these probabilities; conditioned on the random variables generated by the Beta distributions, the indicator variables for edges between pairs are mutually independent.

1.4 The Organization of the Paper

The section 2 introduces the notions of compatibility, layering of a DAG, minimal layering of a DAG and the skeleton for a given classification and ordering of the classes. The skeleton is a DAG with one compelled node to every node in a higher rank class j from a node in the class with rank $j-1$. The class with lowest rank = 1 has no parents. This locks the generation scheme so that a DAG and its minimal layering is the outcome, which is a crucial difference with the edge generation scheme of Kemp et al..

In section 3 we find the required prior probabilities for the components of the minimal layering. We recapitulate an explicit expression for the minimal Hoppe-Beta prior by applying the properties of skeletons.

In a special case, the Hoppe urn scheme involves only three hyperparameters, which may be used to control explicitly the sparsity and consistency of the DAGs generated; these notions will be defined and analysed in section 4 below.

The minimal Hoppe-Beta prior, denoted by $\mathbb{P}_{\mathcal{G}}(G)$, is invoked to compute the posterior probability probability $\mathbb{P}_{\mathcal{G}|\mathbf{X}}(G|\mathbf{x})$ by

$$\mathbb{P}_{\mathcal{G}|\mathbf{X}}(G|\mathbf{x}) \propto \mathbb{P}_{\mathcal{G}}(G)\mathbb{P}_{\mathbf{X}|\mathcal{G}}(\mathbf{x}|G). \tag{2}$$

Here the likelihood $\mathbb{P}_{\mathbf{X}|\mathcal{G}}(\mathbf{x}|G)$ of G invoked is the Cooper-Herskovits marginal data likelihood (or predictive probability) of $\mathbf{x}$ as established in [2].

To find the Maximum A Posteriori Structure (MAPS), maximizing $\mathbb{P}_{\mathcal{G}|\mathbf{X}}(G|\mathbf{x})$, a stochastic optimisation algorithm, as given in section 5, is used. The moves are based on the Hoppe-Ewens urn scheme for moving between classes, along with some addition / deletion of edges.

In section 6 inference from the posterior for three prior distributions is compared by simulations: the uniform prior over graph structures, the Hoppe-Beta distribution, where we consider the DAG structure and the minimal Hoppe-Beta prior. The moves are constructed so that the minimal layering property is preserved after each move.

2 The Block Structure and Layerings of DAGs

Let $X = (X_1, \ldots, X_d)$ denote d nodes of a graph. Each node belongs to one and only one *class* amongst $\{C_1, C_2, \ldots, C_K\}$. The assignment of nodes to classes is unknown and the number of classes, K, is unknown, too. Let $\mathbf{z} = (z_1, z_2, \ldots, z_d)$ be the *class assignment vector*, where $z_i = j$ means that variable i is of class C_j; the classes are labelled by the positive integers. For example, HEPAR II network has $\mathbf{z} = (1,1,1,1,2,2,2,2,3,3,3,3)$.

For the *ordered block model*, a DAG represents direct influences between the nodes. The nodes are also classified and all the arrows of the DAG are *from* nodes of a lower class C_i *to* nodes of a higher class C_j, $i < j$. The classes represent a hierarchical structure. Therefore, the assignment vectors $\mathbf{z}^{(1)} = (1,2,2) = \{\{1\}_1, \{2,3\}_2\}$ and $\mathbf{z}^{(2)} = (2,1,1) = \{\{2,3\}_1, \{1\}_2\}$ give *different* classification structures. We shall in the sequel call the sets in a partition layers or classes. We denote a permutation of d elements in the Cauchy 2-line notation. E.g. the permutation ρ defined by $\rho(1) = 2$, $\rho(2) = 3$, $\rho(3) = 4$, $\rho(4) = 1$, is denoted by $\left(\begin{smallmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{smallmatrix}\right)$.

We shall formalize the ordered block model by means of some first concepts from *hierarchical graph drawing*, a branch of computer science, that strives for the best possible visual understanding of hierarchical relations. There are several algorithms for achieving this, as surveyed in Tamassia [17]. Hierarchical graph drawing is also known as *layering* of a DAG c.f. Healy and Nikolov [4].

We start with the notion of compatibility:

Definition 2.1. *Let $G = (V, D)$ be a DAG. An* **edge** *$e = (x, y) \in D$ is said to be* **compatible with the class assignment vector z and ordering of classes** *ρ if $(x, y) \in D \implies \rho(z_x) < \rho(z_y)$.*

In words, any edge in D points from a node in class of lower rank to a class of higher rank.

Definition 2.2. *Let $G = (V, D)$ be a* **DAG**. *Then G is said to be* **compatible with the class assignment z and ordering of classes** *ρ, if all edges in D are compatible with the class assignment vector $\mathbf{z}$ and ordering of classes ρ.*

Then we can formulate the concept of layering of a DAG.

Definition 2.3. *Let $V = \{1, 2, \ldots, d\}$ and $G = (V, D)$ be a DAG. Let $\mathbf{z}$ be a class assignment of V, and ρ be an ordering of the classes. A* **layering** *$l(G)$ of G is a DAG*

$$l(G) = l(G, \mathbf{z}, \rho) \stackrel{\text{def}}{=} (V, D, \mathbf{z}, \rho),$$

such that $l(G)$ is compatible with the class assignment $\mathbf{z}$ and the ordering of the classes ρ.

It is clear that every *DAG* has at least one layering. The layering of a DAG is not necessarily unique. We denote the set of DAGs compatible with class assignment $\mathbf{z}$ and ordering of classes ρ by $\mathcal{P}_{\mathbf{z},\rho}$ (this symbol recurs only once, in section 5).

This is illustrated by the DAG in Figure 2. There are three possibilities for layering this DAG:

1. Layer 1 (= C_1): nodes 1 and 2, Layer 2 (= C_2): node 3.

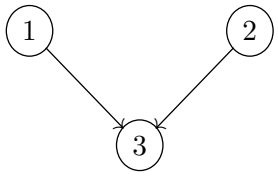

Figure 2: DAG with three nodes

2. Layer 1 (= C_1): node 1, Layer 2 (= C_2): node 2, Layer 3 (= C_3): node 3.

3. Layer 1 (= C_1): node 2, Layer 2 (= C_2): node 1, Layer 3 (= C_2): node 3.

While node 3 is always in the highest layer, there is some freedom of choice with nodes 1 and 2. They can be either in separate layers or in the same layer without violating compatibility.

When inferring classification, the DAG structure alone cannot distinguish between different layerings. For our prior over graph structure, we assume that the layering is minimal in the sense defined next.

Definition 2.4. *Let $G = (V, D)$ be a DAG. The layering $l_{\min}(G)$ of G is said to be* **minimal** *if*

1. *it has the smallest possible number of layers for the DAG G under consideration and*

2. *among layerings satisfying this criterion, the nodes are in layers with as low a rank as possible.*

By this definition, a node in class C_i has at least one parent in class C_{i-1}, because if this was not so, the class C_i would have been pushed downwards to a lower rank. The minimal layering represents, in some sense, the unique ordered class structure that can be compatibly deduced from the DAG and from data. For the three-variable DAG of Figure 2, its minimal layering is the first of the possibilities listed above. A minimal layering can be found in linear time by the longest-path algorithm [ES90] [5, p.419].

Example 2.5. *Let $V = \{1, 2, \ldots, d\}$, $D = ((1, i))_{i=2}^{d}$, $K = 2$, $\mathbf{z} = \left(1, \underbrace{2, 2, \ldots, 2}_{d-1 \text{ slots}}\right)$ and $\rho(1) < \rho(2)$. Then $(V, D, \mathbf{z}, \rho)$ is the minimal layering of a DAG known as the naive Bayes classifier.*

Example 2.6. *Let $V = \{1, 2, \ldots, d\}$, $D = ((i, d))_{i=1}^{d-1}$, $K = 2$, $\mathbf{z} = \left(\underbrace{1, 1, \ldots, 1}_{d-1 \text{ slots}}, 2\right)$ and $\rho(1) < \rho(2)$. Then $(V, D, \mathbf{z}, \rho)$ is the minimal layering of a DAG often referred to as a causal independence model [6]. The minimal layering of the DAG in the Figure 2 is a special case of this with $d = 3$.*

Our technique of generation of DAGs from the minimal Hoppe-Beta prior requires the following graph, too.

Definition 2.7. *Let V be a set of nodes with a classification vector $\mathbf{z} = (z_1, \ldots, z_d)$ for ordered classes $\{C_i\}_{i=1}^K$ ($\rho(i) = i$). A* **skeleton** *$sk\,(V, \mathbf{z}, \rho)$ is a DAG such that for each $i > 1$ and for each node in the class C_i, there is a parent node in class C_{i-1} and each node has no other parents. The edges in the skeleton are called* **compelled edges**.

The minimal layering in Example 2.5 is a skeleton. We have $sk\,(V, \mathbf{z}, \rho) = (V, D_c)$, where D_c stands for compelled directed edges. Any skeleton $sk(V, \mathbf{z}, \rho)$ is obviously a minimal layering of itself in the sense of the Definition 2.4. Clearly, if we know a skeleton for a set of nodes, we can uniquely find the corresponding classes as well as their ordering. There are, in general, several compatible extensions of any skeleton, i.e., DAGs compatible with $\mathbf{z}, \rho$ and with $sk\,(V, \mathbf{z}, \rho)$ as a subgraph.

In the current setting a prior probability distribution for DAGs expresses our beliefs about the corresponding minimal layerings before seeing the data. Hence we are taking

$$\mathbb{P}_{\mathcal{G}}(G) = \mathbb{P}_{\mathcal{G}}\left(l_{\min}(G, \mathbf{z}, \rho)\right). \tag{3}$$

3 The Minimal Hoppe-Beta Prior over Graph Structures

We can express the prior probability (3) by

$$\mathbb{P}_{\mathcal{G}}\left(l_{\min}(G, \mathbf{z}, \rho)\right) = \sum_{sk(V,\mathbf{z},\rho)} \mathbb{P}(l_{\min}(G, \mathbf{z}, \rho) | sk\,(V, \mathbf{z}, \rho)) \mathbb{P}(sk\,(V, \mathbf{z}, \rho)). \tag{4}$$

This involves taking a DAG $G = (V, D)$, finding its minimal layering and then assigning coherently respective prior probabilities to the non-compelled edges $D \setminus D_c$, $sk\,(V, \mathbf{z})$, $\mathbf{z}$ and to ρ. The outline is as follows.

With the prior probabilities established we shall find by (4) the prior probability $\mathbb{P}_{\mathcal{G}}\left(l_{\min}(G, \mathbf{z}, \rho)\right)$, to be called the *minimal Hoppe-Beta Prior*. We first generate a class assignment vector $\mathbf{z}$ according to the Hoppe-Ewens urn scheme to be recapitulated below. Then we give the classes a random ordering ρ. We then ensure that this structure provides the minimal layering for a DAG the sense of Definition 2.7.

For this we choose for every node in C_i one and only one parent node at random (each with equal probability) from the class C_{i-1} and add a corresponding compelled edge. This gives us a skeleton graph. Then we decide on any remaining edges using the same scheme as Kemp et al. with the Hoppe-Beta prior, here refined by some more convenient features. Next the details are provided.

3.1 Class Assignment Vector Generated by Hoppe-Ewens Urn

Firstly, node 1 is assigned to class 1. Assume that nodes $1, \ldots, j$ have been assigned to classes and that there are now a total of k_j different classes. For $(z_1, \ldots, z_d) \in \{1, \ldots, d\}^d$ we set

$$m_i^{(j)}(z_1, \ldots, z_d) = \sum_{l=1}^{j} \mathbf{1}_i(z_l),\ i = 1, \ldots, d,$$

where

$$\mathbf{1}_i(x) = \begin{cases} 1, & x = i \\ 0, & x \neq i. \end{cases}$$

Set

$$m_i^{z;j} \stackrel{\text{def}}{=} m_i^{(j)}(z_1, \ldots, z_d).$$

Thus $m_i^{z;j}$ counts the number of nodes from $\{1, \ldots, j\}$ in class C_i, so that $\sum_{i=1}^{k_j} m_i^{z;j} = j$.

The assignment vector $(z_1, \ldots, z_d)$ generated by the Hoppe-Ewens Urn is the outcome of a random vector $(Z_1, \ldots, Z_d)$. The algorithm begins with $z_1 = 1$ and proceeds as follows: For $j \in \{1, \ldots, d-1\}$,

$$\begin{cases} \mathbb{P}_{Z_{j+1}|Z_1,\ldots,Z_j}(k_j + 1|z_1, \ldots, z_j) = \dfrac{\alpha}{\alpha + j} \\ \mathbb{P}_{Z_{j+1}|Z_1,\ldots,Z_j}(i|z_1, \ldots, z_j) = \dfrac{m_i^{z;j}}{\alpha + j} \qquad i = 1, \ldots, k_j. \end{cases} \tag{5}$$

Here one notes the rule for creating new classes.

3.2 The Probability of the Class Assignment Generated by Hoppe-Ewens Urn

The probability function $\mathbb{P}_Z(z)$ of the assignment vector Z is given by the following well known theorem found in [7]. We need to define the space of outcomes z of Z:

$$\mathcal{S}_d = \{z \in \{1, \ldots, d\}^d | z_1 = 1, 1 \leq z_j \leq \max_{1 \leq k \leq j-1} z_k + 1 : 2 \leq j \leq d\}.$$

We set

$$m_k \stackrel{\text{def}}{=} m_k^{z:d},$$

which is the **occupation number** for class k after all d nodes have been assigned to classes. Then we have the following result.

Theorem 3.1. *Let K be the number of non-empty classes and*

$$\mathcal{S}_d = \left\{ z \in \{1, \ldots, d\}^d | z_1 = 1, 1 \leq z_j \leq \max_{1 \leq k \leq j-1} z_k + 1 : 2 \leq j \leq d \right\}.$$

that is $\mathcal{S}_d$ is the set of possible values of z. The distribution of the random vector Z for a given parameter value $\alpha \in \mathbb{R}_+$ may be computed explicitly and is given by

$$\mathbb{P}_Z(z|\alpha) = \begin{cases} \alpha^K \dfrac{\Gamma(\alpha)}{\Gamma(d + \alpha)} \displaystyle\prod_{k=1}^{K} (m_k - 1)!, & K = 1, \ldots, d, z \in \mathcal{S}_d \\ 0, & \textit{otherwise.} \end{cases} \tag{6}$$

Proof. The proof follows from (5) by the chain rule of probability. □

In the sequel, we shall for ease of writing drop the dependence on the hyperparamter α in $\mathbb{P}_Z(z|\alpha)$.

3.3 Generating DAGs from of The Minimal Hoppe-Beta Prior

For the minimal Hoppe-Beta Prior, we use the Hoppe-Ewens urn scheme to generate the classification.

- **Step 1** For nodes $1, \ldots, d$, generate a class assignment vector according to the Hoppe-Ewens urn scheme. Using $m_i^{(j)} = \sum_{k=1}^{j} \mathbf{1}_i(z_k)$ and let $K_j = \max\{i : m_i^{(j)} \neq 0\}$

$$\mathbb{P}(Z_j = i | Z_1, \ldots, Z_{j-1}) = \begin{cases} \dfrac{m_i^{(j-1)}}{j-1+\alpha} & i = 1, \ldots, K_{j-1} \\ \dfrac{\alpha}{j-1+\alpha} & i = K_{j-1}+1. \end{cases}$$

- **Step 2** Let $K = K_d$. This is the total number of classes. Let ρ be a randomly chosen permutation of $(1, \ldots, K)$; conditioned on K classes, each ρ is chosen with probability $\dfrac{1}{K!}$. The permutation represents the ordering or ranking of the layers, from lowest to highest.

- **Step 3** Let K be the number of classes. For each $j = 2, \ldots, K$, for each node $v \in C_{\rho(j)}$ (where C_i denotes class i) choose a node w randomly from those in class $C_{\rho(j-1)}$ (each with equal probability). Add in the directed edge $w \to v$.

  After stage 3, a skeleton graph has been produced. This is a graph whose minimal layering corresponds to the classification generated by z and ρ and a graph with the minimal number of edges necessary to have this property. The probability of any skeleton graph thus obtained is

$$\mathbb{P}(sk\,(V, \mathbf{z}, \rho)) = \prod_{j=2}^{K} \left(\frac{1}{m_{j-1}} \right)^{m_j}. \tag{7}$$

- **Step 4** The remaining edges which are not in the skeleton are added according to the scheme outlined by Kemp; for $\rho(i) < \rho(j)$, an edge probability ξ_{ij} is generated according to a Beta distribution with parameters $\beta_{1:\rho(i),\rho(j)}$, and $\beta_{2:\rho(i),\rho(j)}$. This is the edge probability between class i nodes and class j nodes, when ρ is the class permutation. This means in other words that we select an edge by a Bernoulli trial with the success probability ξ_{ij}.

  The random variables ξ_{i_1,j_1} and ξ_{i_2,j_2} are independent for $(i_1, j_1) \neq (i_2, j_2)$, $\rho(i_1) < \rho(j_1)$ and $\rho(i_2) < \rho(j_2)$.

This scheme outputs a DAG with a minimal layering (and the order of the classes).

Example 3.2.

In Figure 3a-Figure 3d a possible outcome of the algorithm above is shown. The setup is similar to that in Example 3.3 with $d = 8$,

$$z = \begin{pmatrix} 1 & 1 & 2 & 3 & 3 & 3 & 2 & 1 \end{pmatrix}, \quad \rho = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}.$$

The additional step in this example is in Figure 3c where the skeleton, coloured in red is created.

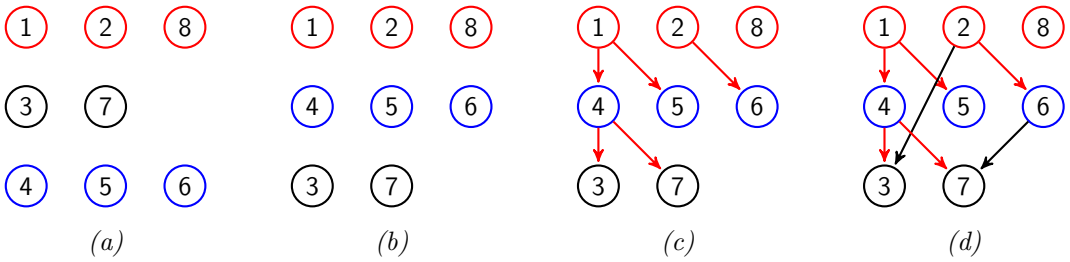

Figure 3: Figures a)-d) show the steps in the minimal Hoppe-Beta prior

3.4 The Edge Probabilities

When constructing the DAG, we only permit edges from nodes in a class a to nodes in a class b if $\rho(a) < \rho(b)$. Conditioned on the assignment vector $\mathbf{z}$ and the class ordering ρ, edges are mutually independent of each other in **Step 4** above. Firstly, we randomly generate the edge probability and then, for each pair, the existence of an edge is the outcome of a Bernoulli random variable with the probability of success equalling the edge probability. For this, let $B(\cdot,\cdot)$ be the **beta function** defined by:

$$B(x,y) = \frac{\Gamma(x)\,\Gamma(y)}{\Gamma(x+y)},$$

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

For $(a,b) \in \{1,\ldots,K\}^2$, define the density $f_{a,b}$ as follows:

$$f_{a,b}(x) = \begin{cases} \dfrac{1}{B(\beta_{1;a,b},\beta_{2;a,b})} x^{\beta_{1;a,b}-1}(1-x)^{\beta_{2;a,b}-1}, & 0 \le x \le 1 \quad a < b \\ 0, & \text{otherwise} \quad a < b \\ \delta_0(x) & a \ge b \end{cases} \tag{8}$$

where for each $a < b$, $\beta_{1;a,b} > 0$ and $\beta_{2;a,b} > 0$. If $\beta_{1;a,b} = 0$ then $f_{a,b} = \delta_0$ and if $\beta_{2;a,b} = 0$ then $f_{a,b} = \delta_1$. $\delta_0(x)$ denotes the unit probability mass at 0.

Let $\eta = (\eta_{a,b} : a,b \in \{1,\ldots,K\})$ be a random matrix of edges probabilities where $\eta_{a,b}$ are independent random variables and, for each $(a,b) \in \{1,\ldots,K\}^2$, $\eta_{a,b} \sim f_{a,b}$.

Let Ξ denote the $d \times d$ matrix with entries $\xi_{i,j} = \eta_{\rho(z_i),\rho(z_j)}$. Then we have the conditional density

$$\pi_{\Xi|Z}(\xi \mid \mathbf{z}) = \prod_{a,b} f_{a,b}\left(\eta_{a,b}\right)$$

$$= \prod_{1\le a<b\le K} \frac{1}{B(\beta_{1;a,b},\beta_{2;a,b})} x^{\beta_{1;a,b}-1}(1-x)^{\beta_{2;a,b}-1} \prod_{1\le b<a\le K} \delta_0\left(\eta_{a,b}\right). \tag{9}$$

Let D denote the edge set of the graph. This is the $d \times d$ matrix with entries

$$d_{ij} = \begin{cases} 1 & \text{edge } i \to j \text{ present} \\ 0 & \text{edge } i \to j \text{ not present} \end{cases}$$

Conditioned on the matrix Ξ, D is the outcome of a random matrix $\mathcal{D}$, where the entries of $\mathcal{D}$ are independent and $\mathcal{D}_{ij} \sim \text{Be}(\xi_{ij})$ (i.e. a Bernoulli trial with success probability ξ_{ij}). It is clear from the construction that D is the edge set of a DAG.

Then the (prior) probability of the DAG G given an assignment vector $\mathbf{z}$, class permutation ρ and the edge probabilities ξ is found by:

$$\mathbb{P}_{\mathcal{G}|\Xi,Z,R}(G|\xi,\mathbf{z},\rho) = \mathbb{P}_{\mathcal{G}|\Xi}(G|\xi) = \prod_{x=1}^{d}\prod_{y=1}^{d}\xi_{x,y}^{d_{x,y}}(1-\xi_{x,y})^{1-d_{x,y}}. \tag{10}$$

with the convention $0^0 = 1$. The DAG thus generated is by construction compatible with the given $\mathbf{z}$ and ρ. Hence $\mathbb{P}_{\mathcal{G}|\Xi,Z,R}(G|\xi,\mathbf{z},\rho) = \mathbb{P}_{\mathcal{G}|\Xi,Z,R}(l(G)|\xi,\mathbf{z},\rho)$.

There are two computationally convenient settings, which give flexibility: $\beta_{i;j,j+1} = \beta_{i,1}$ for $i = 1,2$ and $\beta_{i;j,j+k} = \beta_{i,2}$ for $k \geq j+2$, $i = 1,2$. The parameters are chosen so that the edge probability from j to $j+1$ is higher than j to $j+k$ where $k \geq 2$.

Example 3.3.

In Figure 4a-Figure 4d a possible outcome of the algorithm is shown in 4 steps. Here $d = 8$,

$$z = (1\,1\,2\,3\,3\,3\,2\,1), \quad \rho = \left(\begin{smallmatrix}1&2&3\\1&3&2\end{smallmatrix}\right).$$

Figure 4a shows the nodes. Figure 4b shows the partition of the nodes generated by Hoppe's urn scheme, where the colours indicate the different classes in the partition. Class 1 is coloured in red, class 2 is coloured in black and class 3 is coloured in blue. Figure 4c, shows the new order of the classes, ρ. Figure 4d shows a possible graph under these conditions.

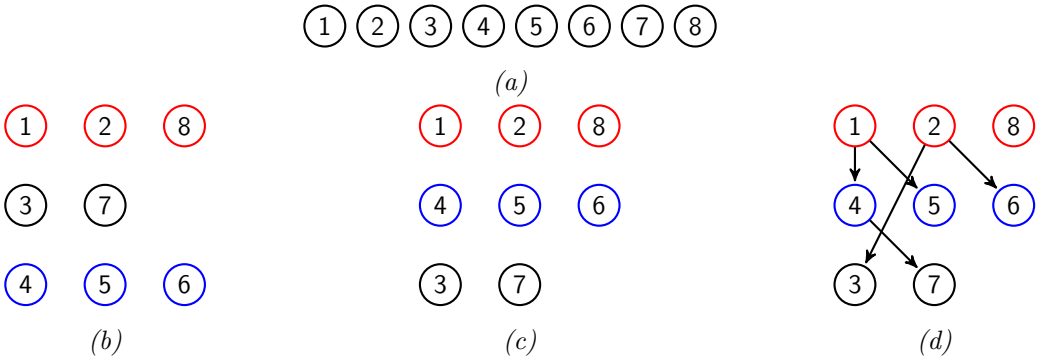

Figure 4: Figures a)-d) show the steps in Example 3.3

In order to proceed we derive the generic distribution of the edges given the classification Assignment and ordering

3.5 The Bayes Integrals of Edge Probabilities

Let Z be the random variable having classification assignment vectors as values and $\mathcal{G}$ a random variable with DAGs as values. Let us for the moment suppress the distinction between compelled and non-compelled edges. We proceed by marginalising over Ξ in $\mathbb{P}_{\mathcal{G},\Xi|R,Z} = \pi_{\Xi|Z,R}\mathbb{P}_{\mathcal{G}|\Xi,Z}$ gives the following Bayes integral:

Lemma 3.4. *Let G be a DAG. Then for any of its layerings $l(G)$*

$$\mathbb{P}_{\mathcal{G}|R,Z}(l(G)|\rho,\mathbf{z}) = \prod_{1\le\rho(a)<\rho(b)\le K} \frac{B(\beta_{1;\rho(a),\rho(b)}+N_{a,b},\beta_{2;\rho(a),\rho(b)}+M_{a,b})}{B(\beta_{1;\rho(a),\rho(b)},\beta_{2;\rho(a),\rho(b)})}, \tag{11}$$

where $N_{a,b}$ is the number of edges between class a and class b and $M_{a,b}$ is the corresponding number of missing edges.

Proof. We use first (10).

$$\begin{aligned}\mathbb{P}_{\mathcal{G}|R,Z}(l(G)|\rho,\mathbf{z}) &= \int_\xi \mathbb{P}_{\mathcal{G}|\Xi,Z}(l(G)|\xi,\mathbf{z})\pi_{\Xi|R,Z}(\xi|\rho,z)d\xi \\ &= \int_\xi \prod_{x=1}^{d}\prod_{y=1}^{d} \xi_{x,y}^{d_{x,y}}(1-\xi_{x,y})^{1-d_{x,y}}\pi_{\Xi|R,Z}(\xi|\rho,\mathbf{z})d\xi.\end{aligned}$$

Next we use (9) and let $N_{a,b}$ be the number of existing edges between class a and class b (provided $\rho(a)<\rho(b)$) and $M_{a,b}=m_am_b-N_{a,b}$ be the number of missing edges. m_a and m_b denote the numbers of nodes in classes a and b respectively. Recall that $\xi_{i,j}=\eta_{\rho(z_i),\rho(z_j)}$. Then it follows by properties of the Beta integral

$$\begin{aligned}&= \prod_{1\le\rho(a)<\rho(b)\le K} \int_0^1 \eta_{\rho(a),\rho(b)}^{N_{a,b}}(1-\eta_{\rho(a),\rho(b)})^{M_{a,b}} f_{\rho(a),\rho(b)}(\eta_{\rho(a),\rho(b)})d\eta_{\rho(a),\rho(b)} \\ &= \prod_{1\le\rho(a)<\rho(b)\le K} \int_0^1 \eta_{\rho(a),\rho(b)}^{N_{a,b}}(1-\eta_{\rho(a),\rho(b)})^{M_{a,b}} \frac{\eta_{\rho(a),\rho(b)}^{\beta_{1;\rho(a),\rho(b)}-1}(1-\eta_{\rho(a),\rho(b)})^{\beta_{2;\rho(a),\rho(b)}-1}}{B(\beta_{1;\rho(a),\rho(b)},\beta_{2;\rho(a),\rho(b)})}d\eta_{\rho(a),\rho(b)} \\ &= \prod_{1\le\rho(a)<\rho(b)\le K} \int_0^1 \frac{\eta_{\rho(a),\rho(b)}^{\beta_{1;\rho(a),\rho(b)}-1+N_{a,b}}(1-\eta_{\rho(a),\rho(b)})^{\beta_{2;\rho(a),\rho(b)}-1+M_{a,b}}}{B(\beta_{1;\rho(a),\rho(b)},\beta_{2;\rho(a),\rho(b)})}d\eta_{\rho(a),\rho(b)} \\ &= \prod_{1\le\rho(a)<\rho(b)\le K} \frac{B(\beta_{1;\rho(a),\rho(b)}+N_{a,b},\beta_{2;\rho(a),\rho(b)}+M_{a,b})}{B(\beta_{1;\rho(a),\rho(b)},\beta_{2;\rho(a),\rho(b)})}.\end{aligned}$$

□

3.6 The Expression for the Minimal Hoppe-Beta Prior

Theorem 3.5. *Let G be a DAG on d nodes whose minimal layering has K layers, with $m_1,\dots,m_K$ in each, the layers ordered from lowest to highest. The generation scheme* **Step 1** - **Step 4** *above gives the (prior) probability of the DAG as:*

$$\begin{aligned}\mathbb{P}_{\mathcal{G}}(G) &= \frac{1}{K!}\frac{\alpha^K\prod_{i=1}^K(m_i-1)!}{\prod_{j=1}^d(\alpha+j-1)}\prod_{j=1}^{K-1}\frac{B(\beta_{1;j,j+1}+N_{j,j+1}-m_{j+1},\beta_{2;j,j+1}+M_{j,j+1})}{B(\beta_{1;j,j+1},\beta_{2;j,j+1})} \\ &\quad\times\prod_{j=1}^{K-2}\prod_{i=j+2}^{K}\frac{B(\beta_{1;j,i}+N_{j,i},\beta_{2;j,i}+M_{j,i})}{B(\beta_{1;j,i},\beta_{2;j,i})}.\end{aligned} \tag{12}$$

Here $N_{a,b}$ denotes the number of edges between class a and class b nodes, while $M_{a,b}=m_am_b-N_{ab}$ denotes the number of missing edges w.r.t. to a complete graph.

Proof. Let is begin by considering

$$P_1 \stackrel{\text{def}}{=} \prod_{j=1}^{K-1} \frac{B(\beta_{1;j,j+1} + N_{j,j+1} - m_{j+1}, \beta_{2;j,j+1} + M_{j,j+1})}{B(\beta_{1;j,j+1}, \beta_{2;j,j+1})} \times \prod_{j=1}^{K-2} \prod_{i=j+2}^{K} \frac{B(\beta_{1;j,i} + N_{j,i}, \beta_{2;j,i} + M_{j,i})}{B(\beta_{1;j,i}, \beta_{2;j,i})}.$$

We introduce the auxiliary P_1 for ease of reference. By **Step 4** and as in the proof of lemma 3.4 we see that the first factor of P_1 follows from forcing each node of class j to have at least one parent in class $j-1$; the second factor, consisting of a double product) follows because there is no such forcing between different pairs of classes and as in the proof of lemma 3.4.

The factor $\frac{1}{K!}$ is by **Step 2** the prior probability of ρ. The factor $\frac{\alpha^K \prod_{i=1}^{K}(m_i-1)!}{\prod_{j=1}^{d}(\alpha+j-1)}$ is the prior probability of $\mathbf{z}$ given ρ in view of Theorem 3.1 and **Step 1**.

Thus we have the prior probability

$$\mathbb{P}(l_{\min}(G,\mathbf{z},\rho)|sk\,(V,\mathbf{z},\rho)) = \frac{1}{K!}\frac{\alpha^K \prod_{i=1}^{K}(m_i-1)!}{\prod_{j=1}^{d}(\alpha+j-1)} P_1. \tag{13}$$

But here we note that this prior probability is the same for all skeletons $sk\,(V,\mathbf{z},\rho)$. Hence we have in (4)

$$\mathbb{P}_{\mathcal{G}}\,(l_{\min}(G,\mathbf{z},\rho)) = \frac{1}{K!}\frac{\alpha^K \prod_{i=1}^{K}(m_i-1)!}{\prod_{j=1}^{d}(\alpha+j-1)} P_1 \times \sum_{sk(V,\mathbf{z},\rho)} \mathbb{P}(sk\,(V,\mathbf{z},\rho)), \tag{14}$$

and the claimed result follows as asserted. □

The prior probability in the theorem can be evaluated for any DAG G. One finds first $l_{\min}(G)$, computes the class assignment vector and the occupation numbers and inserts in (13). Clearly, those DAG architectures displaying higher prior probabilities are, a priori more eligible to modelling by ordered blocks.

Next we give some examples and make some observations about the nature of the DAG prior in the special case, when its is controlled by three parameters. The possibility for this kind of prior analysis does not emerge frequently in structure learning for DAGs, and is due to the well known properties of the Hoppe-Ewens urn scheme, c.f. [12].

4 Properties of the DAGs under the minimal Hoppe-Beta Prior

We start by some simple special cases of the formula (12).

4.1 Examples

Example 4.1. *The probability of the empty graph is clearly equal to the probability that there is exactly one class. This is:*

$$\mathbb{P}_\alpha(\textit{empty graph}) = \frac{(d-1)!}{\prod_{j=2}^{d}(j-1+\alpha)}.$$

Clearly, as $\alpha \to 0$, $\mathbb{P}_\alpha(\textit{empty graph}) \to \mathbb{P}_0(\textit{empty graph}) = 1$. *In addition, if* $\alpha \to +\infty$, *then* $\mathbb{P}_\alpha(\textit{empty graph}) \to 0$, *as seems reasonable.*

Example 4.2. *Consider the minimum layering of the DAG of naive Bayes classifier in Example 2.5 above. Then the prior probability of the minimum layering of this DAG is*

$$\mathbb{P}_\alpha(\textit{naive Bayes classifier}) = \frac{1}{2}\frac{\alpha^2 d!}{\prod_{j=1}^{d}(\alpha+j-1)},$$

as the minimum layering is also the skeleton and hence all edges are in fact compelled edges from the only node in the first layer to the second layer.

Example 4.3. *Then* $K = 2$, $N_{1,2} = d-1$, $m_1 = d-1$, $m_2 = 1$, $M_{1,2} = (d-1) - (d-1) = 0$ *and by (12) the prior probability of the minimum layering of this DAG is*

$$\mathbb{P}_\alpha(\textit{independent causes}) = \frac{1}{2}\frac{\alpha^2 d!}{\prod_{j=1}^{d}(\alpha+j-1)}\frac{B(\beta_{1;1,2}+d-2,\beta_{2;1,2})}{B(\beta_{1;1,2},\beta_{2;1,2})}.$$

We note that the DAGs in the Examples 4.2 and 4.3 above do not have the same the minimal Hoppe-Beta Prior probability unless $d = 2$. Of course, they both have two classes with occupation numbers $d-1$ and 1. They differ in ranking and in the common direction of the edges. Thus the prior expresses different prior beliefs about the common direction of all of the edges. Of course, the meaning of the DAG with $d-1$ independent causes is quite different from the DAG representing $d-1$ variables that are conditionally independent of each other given the single classification variable.

Example 4.4. *In HEPAR II of Figure 1 the edges are constrained so that they are exclusively from a layer to its nearest successor. For this to happen with certainty, one should constrain the prior in (8) by*

$$f_{1,3}(x) = \delta_0(x).$$

Then we have $K = 3$, $N_{1,2} = N_{2,3} = 12$, $m_1 = 4$, $m_2 = 4$, $m_3 = 4$, $M_{1,2} = M_{2,3} = 16 - 12 = 4$ *and by (12) the prior probability of the minimum layering of HEPAR II is*

$$\mathbb{P}_\alpha(\textit{HEPAR II}) = \frac{1}{6}\frac{\alpha^3 6^3}{\prod_{j=1}^{12}(\alpha+j-1)}\frac{B(\beta_{1;1,2}+8,\beta_{2;1,2}+4)}{B(\beta_{1;1,2},\beta_{2;1,2})}\frac{B(\beta_{1;2,3}+8,\beta_{2;2,3}+4)}{B(\beta_{1;2,3},\beta_{2;2,3})}.$$

Example 4.5. *A simple example of the root cause analysis (RCA) of [18] is found if the nodes* $10, 11, 12$ *and all edges leading to them are eliminated in the Figure 1 (and the edge* $(6, 9)$ *is added). The architecture of this pertinent RCA can be thought of as being obtained by concatenating one layer of lowest rank with* $d = 4$ *below the layer of lowest rank in the independent causes DAG of Example 4.3 with* $d = 5$.

As in the preceding example, we take in (8)

$$f_{1,3}(x) = \delta_0(x).$$

Then we have $K = 3$, $N_{1,2} = 12$,$N_{2,3} = 4$, $m_1 = 4$, $m_2 = 4$, $m_3 = 1$, $M_{1,2} = 4$,$M_{2,3} = 4 - 4 = 0$ *and by (12) the prior probability of the minimum layering of this RCA is*

$$\mathbb{P}_\alpha(RCA) = \frac{1}{6}\frac{\alpha^3 6^2}{\prod_{j=1}^{9}(\alpha+j-1)}\frac{B(\beta_{1;1,2}+8,\beta_{2;1,2}+4)}{B(\beta_{1;1,2},\beta_{2;1,2})}\frac{B(\beta_{1;2,3}+3,\beta_{2;2,3})}{B(\beta_{1;2,3},\beta_{2;2,3})}.$$

As expected by construction, the factor of Bayes integrals above is nothing but the product of the Bayes integral of independent causes with $d = 5$ *with the Bayes integral for a layer in HEPAR II (with* $d = 8$*).*

When $\beta_{i;a,b} = \beta_i$, $i = 1, 2$ for all $a < b$, it is possible to compute some reasonably straightforward but instructive properties of the DAGs sampled from the minimal Hoppe-Beta prior. In this case, the prior is a function of three parameters; α, β_1 and β_2.

While leading to computational convenience, removing the dependence of β_1 and β_2 on a and b means that edges are equally likely between a node and any other of a higher order.

Note Consider the DAG in Figure 2. Suppose the partition is $C_1 = \{1\}$, $C_2 = \{2\}$, $C_3 = \{3\}$. If $\beta_{i;a,b} = \beta_i$: $i = 1, 2$ for all $a < b$, then the edge probability $1 \mapsto 3$ is the same whether the classes appear in the order C_1, C_2, C_3 or C_2, C_1, C_3. If $\beta_{i:1,3} \neq \beta_{i:2,3}$, then the order makes a difference.

4.2 Expected number of cells

Let K_d be the number of classes generated by Hoppe's urn scheme with d nodes. Let I_i to be the indicator function of the event that a new class is created in round i, $i = 1, \ldots, d$. Then $\mathbb{P}(I_i = 1) = \mathbb{E}[I_i] = \dfrac{\alpha}{\alpha + i - 1}$ for $i = 1, \ldots, N$. The expected number of classes is:

$$\mathbb{E}[K_d] = \mathbb{E}[\sum_{i=1}^{d} I_i] = \frac{\alpha}{\alpha} + \frac{\alpha}{\alpha + 1} + \cdots + \frac{\alpha}{\alpha + d - 1}.$$

Note that

$$\frac{\alpha}{\alpha} + \frac{\alpha}{\alpha + 1} + \cdots + \frac{\alpha}{\alpha + d - 1} = \alpha \mathcal{H}_d(\alpha)$$

where $\mathcal{H}_d$ is the generalised harmonic number. It is straightforward to compute that for any fixed α, $\mathcal{H}_d(\alpha) \sim \ln(d)$ as $d \to \infty$. Thus

$$\lim_{d \to +\infty} \frac{\mathbb{E}[K_d]}{\ln(d)} = \alpha.$$

Further analytic and probabilistic properties of K_d are found in [12, pp. 176-179].

4.3 Sparsity

Here, the prior is a function of three parameters; α, β_1 and β_2.

One convenient measure of sparsity is to consider the expected number of edges and compare it to the number of edges for a complete DAG on d nodes, which is $\dfrac{1}{2} d(d-1)$.

$$\begin{aligned}
\mathbb{E}[\text{edges}] &= \left(\frac{\beta_2}{\beta_1 + \beta_2}\right)(d - \mathbb{E}\left[m_{\rho(1)}\right]) + \left(\frac{\beta_1}{\beta_1 + \beta_2}\right)\mathbb{E}\left[\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} m_{\rho(i)} m_{\rho(j)}\right] \\
&= \left(\frac{\beta_2}{\beta_1 + \beta_2}\right)(d - \mathbb{E}\left[m_{\rho(1)}\right]) + \frac{1}{2}\left(\frac{\beta_1}{\beta_1 + \beta_2}\right)\left(d^2 - \mathbb{E}\left[\sum_{i=1}^{K} m_i^2\right]\right).
\end{aligned}$$

This comes from the following: firstly, each of the d nodes has at least one parent (a compelled edge) for the skeleton, except for those in the lowest layer. Then there are the remaining edges. For $\rho(j) = \rho(i) + 1$, consider the non-compelled edges which are added in, each with probability

$\frac{\beta_1}{\beta_1+\beta_2}$. For the second line, only nodes within the same block cannot have an edge. Recall that $\mathbb{E}[K] \simeq \alpha \ln(d)$. Now suppose that $\alpha(d)$ varies with d and that $\alpha' := \alpha(d)\ln(d)$ is kept constant. For fixed $\alpha' := \alpha(d)\ln(d) > 0$, can be shown that $\lim_{d\to+\infty} \frac{1}{d^2}\mathbb{E}\left[\sum_{i=1}^{K} m_i^2\right] = f(\alpha')$ where f is a decreasing function defined on $\mathbb{R}_+$ which satisfies $f(0) = 1$ and $f(+\infty) = 0$. It follows that

$$\lim_{d\to+\infty} \frac{\mathbb{E}\left[\text{edges}\right]}{\frac{1}{2}d(d-1)} = \left(\frac{\beta_1}{\beta_1+\beta_2}\right)\left(1 - f(\alpha')\right).$$

This may be considered as a sparsity index, since $\frac{1}{2}d(d-1)$ is the maximum number of possible edges. From this expression, it is clear that there are two parameters for controlling the sparsity. Firstly, low values of $\left(\frac{\beta_1}{\beta_1+\beta_2}\right)$ lead to a sparse graph. Secondly, low values of $\alpha\ln(d)$ lead to a sparse graph.

4.4 Consistency

As discussed in Mansinghka et al., the role of $\beta_1+\beta_2$ is also of interest. While the expected number of edges only depends on β_1 and β_2 only through $\frac{\beta_1}{\beta_1+\beta_2}$, the sum of $\beta_1+\beta_2$ provides a *consistency* parameter. If $\beta_1+\beta_2$ is small (with $\frac{\beta_2}{\beta_1+\beta_2} = 0.5$), the outcome of the random variable ξ_{ij}, as defined in **Step 4** above, which has the Beta distribution with parameters β_1, β_2 will typically take values either close to 0 or close to 1. This prior will therefore generate graphs which either have many edges between a chosen pair of classes or few edges between a chosen pair of classes (while the average edge probability is 0.5). If $\beta_1+\beta_2$ is large, the edge probabilities between chosen pairs of classes will be similar.

5 The Posterior Distribution and A Stochastic Search Algorithm

Now suppose that the d nodes of the graph are random variables $(X_1, \ldots, X_d)$ and an $n \times d$ data matrix $\mathbf{x}$ containing n independent instantiations of the d variables under study (identified as nodes) is given.

Let $\mathbf{X}$ denote the random matrix from which $\mathbf{x}$ is an observation. The aim is to infer a DAG along which the probability distribution factorises and the class structure, which is the minimal layering of the DAG.

5.1 Cooper-Herskovits Likelihood and Posterior

Our likelihood principle is

$$\mathbb{P}_{\mathbf{X}|\mathcal{G}} = \mathbb{P}_{\mathbf{X}|l_{\min}(\mathcal{G})}.$$

The leikelihood may be evaluated explicitly and is well known as the Cooper-Herskovits likelihood, an example of a marginal data likelihood, derived in [2]. For minimal layered graphs it is given by:

$$\mathbb{P}_{\mathbf{X}|\mathcal{G}}(\mathbf{x}|G) = \prod_{r=1}^{K} \prod_{j \in C_r} \prod_{l=1}^{q_j} \frac{\Gamma(\sum_{i=1}^{p_j} \gamma_{jil})}{\Gamma(n(\pi_j^{(l)}) + \sum_{i=1}^{p_j} \gamma_{jil})} \prod_{i=1}^{p_j} \frac{\Gamma(\gamma_{jil} + n(x_j^{(i)}, \pi_j^{(l)}))}{\Gamma(\gamma_{jil})}. \tag{15}$$

Here $(x_j^{(1)}, \ldots, x_j^{(p_j)})$ is the state space for variable X_j, while $(\pi_j^{(1)}, \ldots, \pi_j^{(q_j)})$ is a listing of the possible parent configurations for variable j in the Bayesian network. The parameters $(\gamma_{jil} : j = 1, \ldots, d; i = 1, \ldots, p_j; l = 1, \ldots, q_j)$ are hyperparameters that can be chosen depending on prior information. We take $\gamma_{jil} = \gamma > 0$ all equal, so that there is one free parameter γ for the Cooper-Herskovits likelihood. Thus the choice of hyperparameters does not reflect the layering.

The probability $\mathbb{P}_{\mathcal{G}}$ from (12) together with the Cooper-Herskovits likelihood gives the posterior:

$$\mathbb{P}_{\mathcal{G}|\mathbf{X}}(G|\mathbf{x}) \propto \mathbb{P}_{\mathcal{G}}(G)\mathbb{P}_{\mathbf{X}|\mathcal{G}}(\mathbf{x}|G) =: \mathbb{S}(G|\mathbf{x}) \tag{16}$$

We shall next suggest an algorithm that maximizes $\mathbb{S}(G|\mathbf{x})$ as a function of G. This produces the Maximum A Posterior estimate of the DAG with an ordered block structure.

5.2 A Stochastic Search Algorithm

In the algorithms described below, computational savings are made if only a small part of the graph needs to be considered. Suppose only one edge at a time is changed. Let $G_{ij} = 1$ if there is an edge $i \mapsto j$ and 0 if there is no edge $i \mapsto j$. Let $(x_i^1, \ldots, x_i^{p_i})$ denote the state space of variable X_i. Let G^- denote a DAG where $G_{ij} = 0$ and let G^+ denote the graph G^- with G_{ij} replaced by $G_{ij} = 1$. Assume that G^+ is a DAG. Let q_{j+} denote the number of parent configurations for variable j in G^+ and q_{j-} the number in G^-. Note that $q_{j+} = p_i q_{j-}$. The ratio $\dfrac{\mathbb{P}_{\mathcal{X}|\mathcal{G}}(\mathbf{x}|G^+)}{\mathbb{P}_{\mathcal{X}|\mathcal{G}}(\mathbf{x}|G^-)}$ may be computed, from (15), as:

$$\frac{\mathbb{P}_{\mathcal{X}|\mathcal{G}}(\mathbf{x}|G^+)}{\mathbb{P}_{\mathcal{X}|\mathcal{G}}(\mathbf{x}|G^-)} = \left(\frac{\Gamma(p_j\gamma)}{\Gamma(\gamma)^{p_j}}\right)^{q_{j-}(p_i-1)} \prod_{l=1}^{q_{j-}} \frac{\Gamma(p_j\gamma + n(\pi_j^{-(l)}))}{\prod_{k=1}^{p_j} \Gamma(p_j\gamma + n(\pi_j^{-(l)}, x_i^{(k)}))} \prod_{k=1}^{p_i} \prod_{\alpha=1}^{p_j} \frac{\Gamma(\gamma + n(\pi_j^{-(l)}, x_j^{(\alpha)}))}{\prod_{k=1}^{p_j} \Gamma(p_j\gamma + n(\pi_j^{-(l)}, x_i^{(k)}, x_j^{(\alpha)}))},$$

where $(\pi_j^{-(1)}, \ldots, \pi_j^{-(q_{j-})})$ is an enumeration of the parent configurations of variable j in graph G^-. This formula only depends on the variable X_j, the parent set Pa_j^- of X_j in graph G^- and the additional variable X_i.

We tried two algorithms; a Gibbs sampler and a stochastic optimisation algorithm. These two algorithms have different objectives. The aim of a Gibbs sampler is to generate an empirical distribution which approximates the posterior distribution. This is useful for exploring properties of the posterior. The stochastic optimisation algorithm simply looks for the maximum aposteriori structure.

While the Gibbs sampler is theoretically ergodic, convergence was very slow. The main difficulty was that nodes in the wrong layer had difficulty bubbling up to their correct layer.

With this in mind, the stochastic optimisation algorithm was constructed to ensure mobility between layering. The moves were constructed by choosing a node at random and re-assigning it according to to a Hoppe-Ewens urn scheme.

5.3 Stochastic search algorithm

In this section we present the stochastic search algorithm used for maximising, at least approximately, the posterior score function defined in Equation (16). The algorithm generates a non-reversible Markov chain of DAGs $\{G^{(t)}\}_{t=1}^{T}$, started from an initial DAG $G^{(0)}$. A move from site $G^{(t)}$ to site $G^{(t+1)}$ is then made by proposing a move to G' according to the distribution $Q(G^{(t)};.)$ where the transition kernel Q is described below and then accepting the move with probability

$$\alpha_{G^{(t)},G'} = \min\left\{1\,,\,\frac{\mathbb{S}(G'|\mathbf{x})}{\mathbb{S}(G^{(t)}|\mathbf{x})}\right\}.$$

5.3.1 The Proposal Kernel Q

For each DAG $G^{(t)}$ in the chain we let $\mathbf{z}^{(t)}$ denote the corresponding assignment vector of the minimal layering. The algorithm is initiated with the empty graph $G^{(0)}$, thus $\mathbf{z}^{(0)}$, the assignment vector of $l_{min}(G^{(0)})$ is a vector of zeros. We proceed as follows:

1. Let $\widetilde{\mathbf{z}} = \mathbf{z}^{(t)}$, the assignment vector in $l_{\min}(G^{(t)})$ and call $\widetilde{\mathbf{z}^{(t)}}_{\text{old}} = \widetilde{\mathbf{z}}$. Choose a node x at random and remove it (each node with equal probability). If the chosen node was in a layer of its own, we decrease the order of each layer with order $\geq j$, that is set $\widetilde{\mathbf{z}}_{\text{new}} = \widetilde{\mathbf{z}}_{\text{old}} - 1$ for all $\widetilde{z}_k$ such that $\widetilde{z}_k \geq j$ and then set $\widetilde{\mathbf{z}} = \widetilde{\mathbf{z}}_{\text{new}}$, so that there will be no empty layers.

2. Let K be the number of layers left. Now re-assign the node x in the following way: put it in a new layer, $(K+1)$ with probability $\dfrac{\widetilde{\alpha}}{\widetilde{\alpha}+(d-1)}$ and in layer j for $1 \leq j \leq K$ with probability $\dfrac{m_j}{\widetilde{\alpha}+(d-1)}$ where m_j is the current occupancy number in layer j. In other words, the node is reassigned according to a Hoppe-Ewens scheme with parameter $\widetilde{\alpha} > o$.

3. If x is now in layer $K+1$, let m be chosen randomly according to the uniform distribution over $\{1,\ldots,K+1\}$ and set:

$$\begin{cases} \hat{z}_j = \widetilde{z}_j, & j : 1 \leq \widetilde{z}_j \leq m-1 \\ \hat{z}_i = m, & j : j = i \\ \hat{z}_j = \widetilde{z}_j + 1, & j : m+1 \leq \widetilde{z}_j \leq K \end{cases}$$

   That is, the new layer gets rank m and the ranks from m to K are pushed one place to the right to compensate. This process has generated a new class assignment vector $\hat{\mathbf{z}}$.

4. To construct a proposal G', let $\widetilde{G} = G^{(t)}$. The class assignment vector $\mathbf{z}'$ for $l_{\min}(G')$ will be derived from $\hat{\mathbf{z}}$ constructed above.

   (a) Remove those edges not compatible with $\hat{\mathbf{z}}$.

(b) If $\hat{z}_x \geq 2$, if x does not have any parent in layer $\hat{z}_x - 1$, add a single compelled edge (p, x), where p is randomly chosen from layer $\hat{z}_x - 1$, each with probability $\frac{1}{\hat{z}_x - 1}$.

(c) If $\hat{z}_x = 1$, add x as a parent to each node in $\hat{z}_x + 1$.

(d) Steps (a), (b), (c) have generated a DAG $\widehat{G}$. Let $\hat{\mathbf{z}}_{\text{new}}$ denote the assignment vector $l(\widehat{G})$. This differs from $\hat{\mathbf{z}}$ only if there are children of x in graph $\widetilde{G}$ for which x was the *only* parent in $\widetilde{G}$. For such nodes, $\hat{z}_{\text{new},c} = \max_{p \in \text{Pa}_{\widetilde{G}}(c) \setminus \{x\}} \widetilde{z}_p + 1$. Here $\text{Pa}_{\widetilde{G}}(c) \setminus \{x\}$ denotes the parent set of c in graph $\widetilde{G}$ without x. This process continues recursively for the children of x until every node in the graph is in its minimal layer.

5. Let $\hat{\mathbf{z}}$ denote the current assignment vector ($\hat{\mathbf{z}}$ from step 3 modified by step 4 (d)). Now choose two nodes y and w at random. If $\hat{z}_y \geq \hat{z}_w$, do nothing. Otherwise, toggle the edge between y and w (i.e remove it if exists and add it if it does not exists). Let G' denote the resulting graph.

6. Let $\mathbf{z}'$ be the class assignment vector of $l_{\min}(G')$. If (y, w) was removed, then place w in its minimal layer as in step 4 (d).

This process generates a proposal DAG $l_{\min}(G')$ with assignment vector $\mathbf{z}'$.

Alternatively, the vector $\hat{\mathbf{z}}$ from step 3 could have been taken as coming from the minimal layering of the graph, with step 4(d) replaced by adding in a minimal number of edges to ensure that children c of node x were in the appropriate class. The rejection rate was higher with this approach.

6 Simulations

6.1 Data generation

The simulation studies were made on random samples from the HEPAR II network shown in Figure 1. All the variables are binary and the parameter in the conditional probability tables where independently sampled from a $Beta(0.5, 0.5)$- distribution and were then adjusted so to ensure that they lie in the range $(0.1, 0.9)$.

6.2 Simulation results

The stochastic search algorithm with the three types of priors (uniform, Hoppe-Beta and minimal Hoppe-Beta) was tested on 500 samples from the HEPAR II network shown as an adjacency matrix in Figure 5 [1]. For the minimal Hoppe-Beta and the Hoppe-Beta prior, the hyper parameters were set to $\beta_{1;i,i+1} = 2$, $\beta_{2;i,i+1} = 1$ for $i = 1, \ldots, K-1$ and $\beta_{1;i,j} = 1$, $\beta_{2;i,j} = 2$, for $i = 1, \ldots, K-2$ and $K \geq j > i+1$. This reflects the prior knowledge that between layers i and $i+1$, the graph is denser, while between layer i and j where $j > i+1$ the graph is sparser. For the sparsity parameter we used $\alpha = 1$. For all the three priors we used $\gamma = 1$ and $\widetilde{\alpha} = 1$.

[1] The same study were performed on 10 different datasets showing similar results, for that reason, results from only one representative dataset are considered here.

As a measure of goodness of fit we use the sensitivity (TPR) and specificity (SPC) defined as follows

$$TPR \stackrel{\text{def}}{=} \frac{\textit{Number of edges correctly identified}}{\textit{Number of edges correctly identified + Number of edges falsely rejected}}$$

and

$$SPC \stackrel{\text{def}}{=} \frac{\textit{Total number of edges in the skeleton}}{\textit{Total number of edges in the skeleton + Number of edges wrongly included}}.$$

Here, the skeleton of a directed graph means its undirected version.

We ran 10 trajectories for 5000 iterations for each prior and the results are summarized in Table 1 and in Figure 6-Figure 8. Since the Hoppe-Beta prior is joint prior over partitions and DAGs, or

$$\mathbb{P}_{\mathcal{G},Z}(G,\mathbf{z}) = \frac{g(G,z)}{K!}\prod_{1\le a<b\le K}\frac{B(\beta_1+N_{a,b},\beta_2+M_{a,b})}{B(\beta_1,\beta_2)}\alpha^K\frac{\Gamma(\alpha)}{\Gamma(d+\alpha)}\prod_{k=1}^{K}(m_k-1)!$$

where $g(G,z)$ is the number of orderings ρ of the classes $1,\dots,K$ such that $G \in \mathcal{P}_{\mathbf{z},\rho}$ (notation associated to the definition 2.3). We used this joint score when evaluating the method. The mean and standard error for the TPR and SPC taken over the 100 best scoring graphs among the 10 chains are found in Table 1. As seen the minimal Hoppe-Beta prior shows the best results in terms of SPC and TPR among the methods.

Read from the top, Figure 6 - Figure 8 show in the left columns, the trajectories for the prior, the Cooper-Herskovits likelihood and the scoring function. The black line in each plot is the corresponding function evaluated at the true HEPAR II network. The right column shows heat maps over: the top 100 scoring DAGs among all 10 trajectories, the single top scoring DAG among the 10 trajectories and the top scoring graph in each trajectory. Looking at the heat maps for the top 100 scoring graphs from the 10 chains, we see that we get the clearest results with the minimal Hoppe-Beta prior.

Prior	TPR	SPC
Uniform	0.77 / 0.09	0.79 / 0.07
Minimal Hoppe-Beta	0.85 / 0.10	0.86 / 0.06
Hoppe-Beta	0.80 / 0.19	0.84 / 0.12

Table 1: The results reported are mean values and standard error (mean/std. error) taken over the 100 best scoring DAGs among the 10 trajectories of the stochastic search algorithm ran for 5000 iterations each.

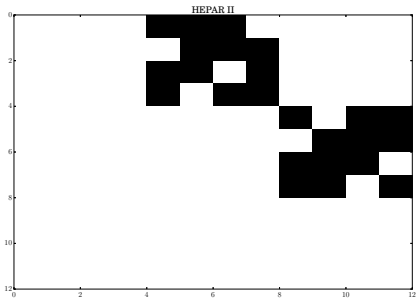

Figure 5: The HEPAR II network shown as an adjacency matrix.

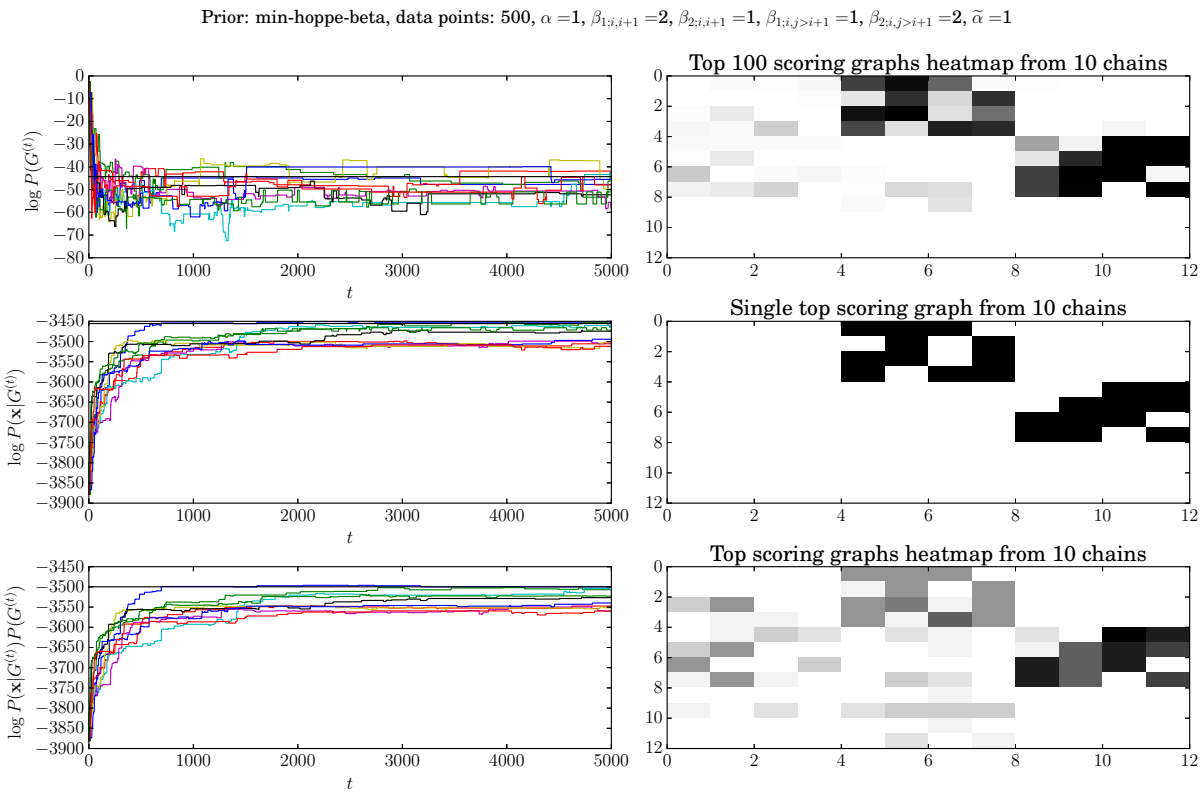

Figure 6: Results from 10 stochastic search trajectories with the minimal Beta-Hoppe prior ran for 5000 iterations each.

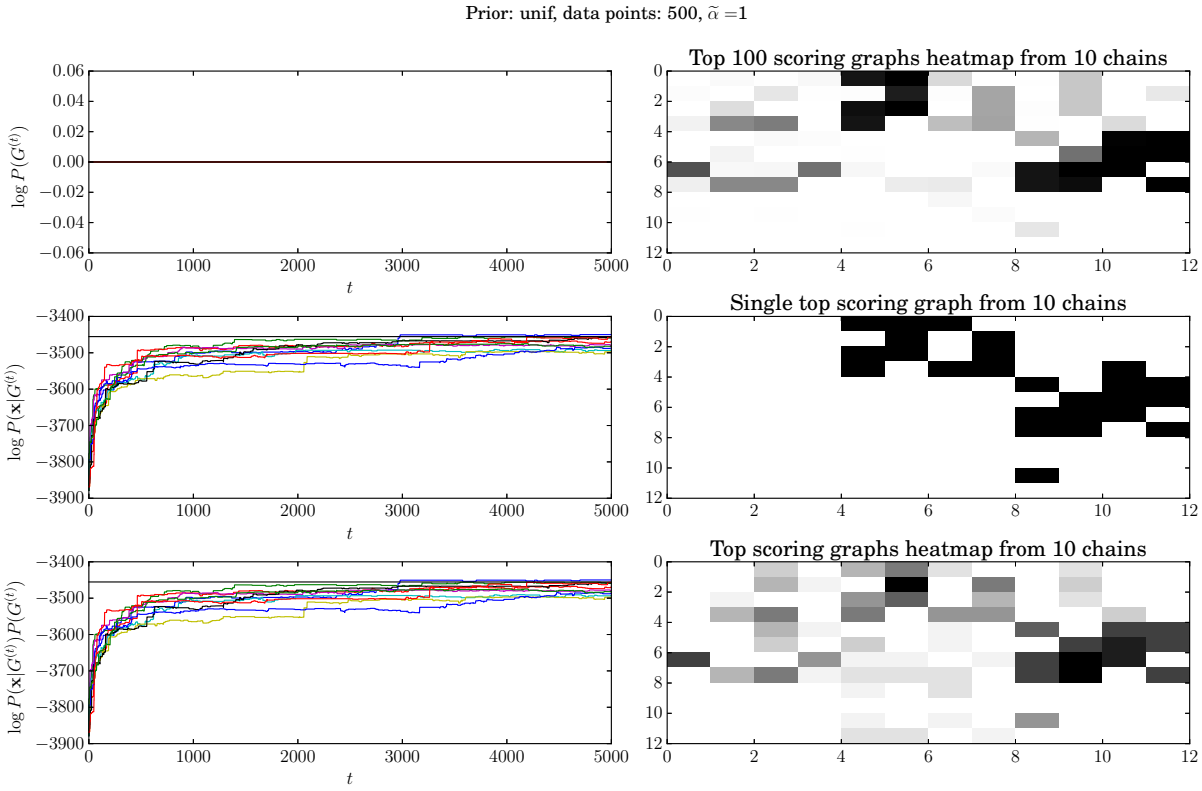

Figure 7: Results from 10 stochastic search trajectories with the uniform prior ran for 5000 iterations each.

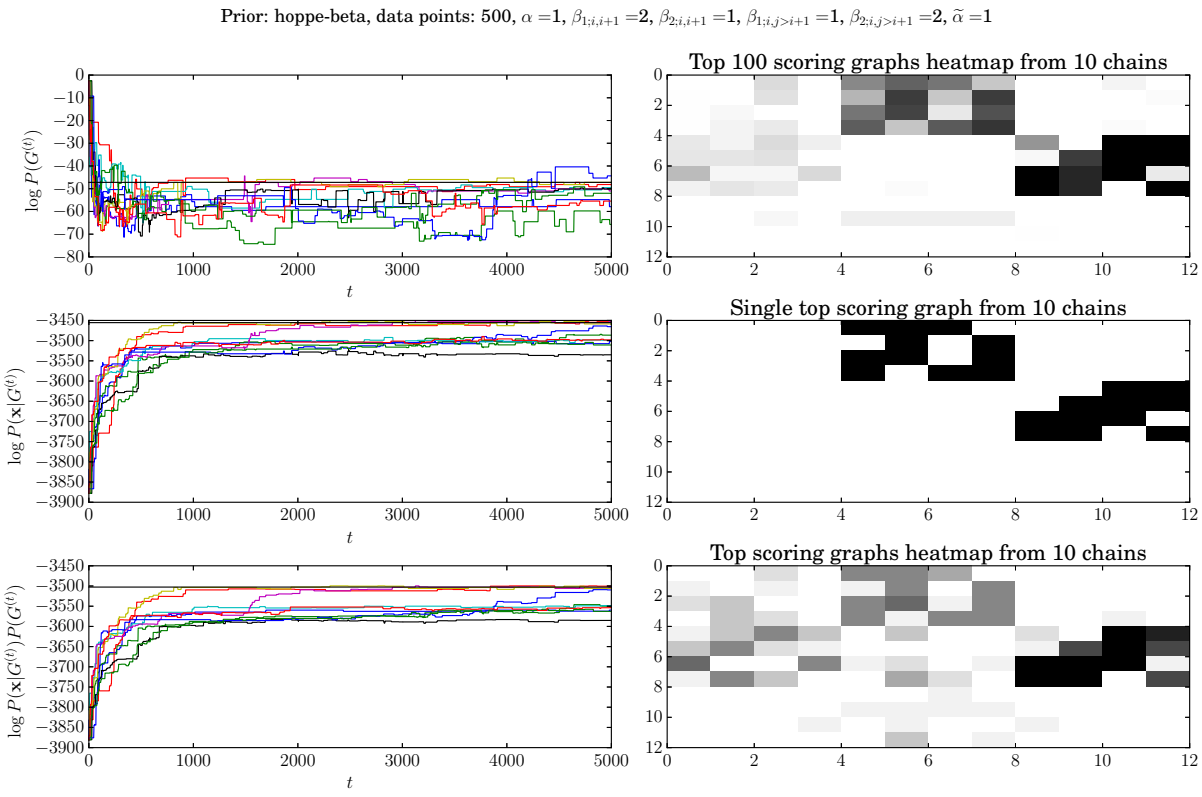

Figure 8: Results from 10 stochastic search trajectories with the Hoppe-Beta prior ran for 5000 iterations each.

7 Summary and Conclusions

This article considered the *Ordered Block Model* introduced by Kemp et al. [8], which was applied to Bayesian Networks for multivariate data by Mansinghka et al. [13]. We introduce a new prior distribution over graph structures, which is a modification of the Hoppe-Beta prior introduced by Kemp et al.. The probability distribution over graphs has an explicit closed form. The parameters can adjusted to determine *sparsity* and *consistency*; consistency refers to the similarity of edge probabilities between nodes of different pairs of classes. We call this prior the *minimal Hoppe-Beta*. It builds on, and represents an advantage over the Hoppe-Beta of Kemp et al., which is a joint prior over graphs and classes. With the minimal Hoppe-Beta, the class structure is implied by the graph.

The prior is then tested experimentally; a posterior is obtained via the Cooper-Herskovits likelihood. We run a stochastic optimisation scheme and compare the output with three different priors; the uniform, the Hoppe-Beta of Kemp et al. and the new Minimal Hoppe-Beta.

The minimal Hoppe-Beta compares favourably.

References

[1] Nicos Angelopoulos and James Cussens. Bayesian learning of bayesian networks with informative priors. *Annals of Mathematics and Artificial Intelligence*, 2008.

[2] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992. ISSN 0885-6125. doi: 10.1007/BF00994110. URL http://dx.doi.org/10.1007/BF00994110.

[3] Stefan Engström. Random acyclic orientations of graphs. trita-mat-e, 2013:02. Technical report, 2013.

[4] Patrick Healy and Nikola Nikolov. How to Layer a Directed Acyclic Graph. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, chapter 2, pages 16–30. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2002.

[5] Patrick Healy and Nikola Nikolov. Hierarchical Drawing Algorithms. In Roberto Tamassia, editor, *Handbook of graph drawing and visualization*, chapter 13, pages 409–453. CRC Press, Baton Roca, 2013.

[6] David Heckerman and John S Breese. Causal independence for probability assessment and inference using bayesian networks. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 26:826–831, 1996.

[7] Fred M.. Hoppe. Polya-like urns and the ewens' sampling formula. *Journal of Mathematical Biology 20: 91.*, 1984.

[8] Charles Kemp, Thomas L. Griffiths, and Joshua B. Tenenbaum. Discovering Latent Classes in Relational Data. Technical report, 2004.

[9] Timo Koski and John Noble. A review of bayesian networks and structure learning. *Mathematica Applicanda*, 40:51–103, 2012.

[10] Jack Kuipers and Giusi Moffa. Uniform random generation of large acyclic digraphs. *Statistics and Computing*, 25(2):227–242, 2013. ISSN 0960-3174. doi: 10.1007/s11222-013-9428-y. URL http://dx.doi.org/10.1007/s11222-013-9428-y.

[11] Jack Kuipers and Giusi Moffa. Partition mcmc for inference on acyclic digraphs. *arXiv preprint arXiv:1504.05006*, 2015.

[12] H. Mahmoud. *Polya Urn Models*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2008. ISBN 9781420059847.

[13] V. K. Mansinghka, C. Kemp, and J. B. Tenenbaum. Structured priors for structure learning. In *In Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI*. AUAI Press, 2006.

[14] Agnieszka Oniśko, Marek J. Druzdzel, and Hanna Wasyluk. Learning bayesian network parameters from small data sets: application of noisy-or gates. *International Journal of Approximate Reasoning*, 27(2):165–182, 2001. ISSN 0888-613X. doi: http://dx.doi.org/10.1016/S0888-613X(01)00039-1. URL http://www.sciencedirect.com/science/article/pii/S0888613X01000391.

[15] R.W. Robinson. Counting unlabeled acyclic digraphs. In CharlesH.C. Little, editor, *Combinatorial Mathematics V*, volume 622 of *Lecture Notes in Mathematics*, pages 28–43. Springer Berlin Heidelberg, 1977. ISBN 978-3-540-08524-9. doi: 10.1007/BFb0069178. URL http://dx.doi.org/10.1007/BFb0069178.

[16] M. A. Shwe, B. Middleton, D. E. Heckerman, M. Henrion, E. J. Horvitz, H. P. Lehmann, and G. F. Cooper. Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base. i. the probabilistic model and inference algorithms. *Methods Archive*, 30(4):241–255, 1991.

[17] Roberto Tamassia. *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)* . Chapman & Hall/CRC, 2008. ISBN 1584884126.

[18] G Weidl, AL Madsen, and S Israelson. Applications of object-oriented bayesian networks for condition monitoring, root cause analysis and decision support on operation of complex continuous processes. *Computers & chemical engineering*, 29(9):1996–2009, 2005.