university of groningen / faculty of science and engineering / mathematics and applied mathematics

Bayesian Coupled Scheme for Network Modelling with Application to the Raf Signalling Pathway

Master Research Project Applied Mathematics

Date: March 20, 2025

Author: R. Rusnák

Student Number: S4508440

First Supervisor: Prof. dr. M.A. Grzegorczyk

Second Supervisor: Prof. dr. J.P. Trapman

Abstract: In many fields involving network modelling, particularly molecular biology, data is often collected under varying experimental conditions. We assume that such data arises from gentle mutations of an underlying network that can be modelled as a Directed Acyclic Graph (DAG). Previous work proposed a Bayesian framework that couples related DAGs through a shared "hypernetwork," effectively capturing features common across all experimental conditions. To not penalize DAGs that encode the same conditional dependencies and independencies, we develop two new coupling mechanisms: one compares DAGs based on their equivalence classes using Completed Partially Directed Acyclic Graphs (CPDAGs), and the other operates with their underlying skeleton structures. Inference is performed via Markov Chain Monte Carlo (MCMC) sampling, and simulation studies demonstrate that our method recovers underlying networks more accurately in a statistically significant manner. A major hurdle in Bayesian network learning is that the number of DAGs grows superexponentially with the number of nodes. This particularly complicates the evaluation of the network prior distribution and forces us to employ approximation techniques. We develop a deterministic mechanism that further enhances the well-known Perfect Gas Approximation, which is often used when the prior distribution over network structures is obtained in the form of a Gibbs distribution. We report a solid improvement in computational efficiency, which allows us to use spare CPU time for additional MCMC iterations and thus further enhance inference accuracy. We demonstrate our model by reconstructing the Raf signalling pathway, an intracellular cascade important in cancer research and drug discovery.

Keywords: Bayesian networks; Bayesian inference; Completed Partially Directed Acyclic Graphs; Raf pathway; Markov Chain Monte Carlo; Perfect Gas Approximation.

Contents

1 Introduction

Raf proteins are a family of signaling proteins that play a crucial role in cell signaling, particularly in the regulation of cell cycle proliferation, survival, division, and responses to stress. The biochemical and functional characterization of Raf proteins was discovered in the 1980s during a study of cancer-causing retroviruses. It was realized that Raf proteins are part of a larger protein network [37] - the Raf signalling pathway, also known as Ras-Raf-MEK-ERK pathway, shown in Figure 1.1. Dysregulation of the Raf pathway leads to tumorigenesis and tumor development [19]. For example, aberrant activation of the Raf pathway significantly contributes to the occurrence and development of the dangerous Hepatocellular carcinoma, the fifth most common tumor [39]. Mutations affecting the Raf pathway are found in up to 33% of all tumors, with approximately 8% being primarily driven by Raf mutations [58, 42]. This makes the study and inference on the Raf pathway especially important in cancer research and drug discovery.

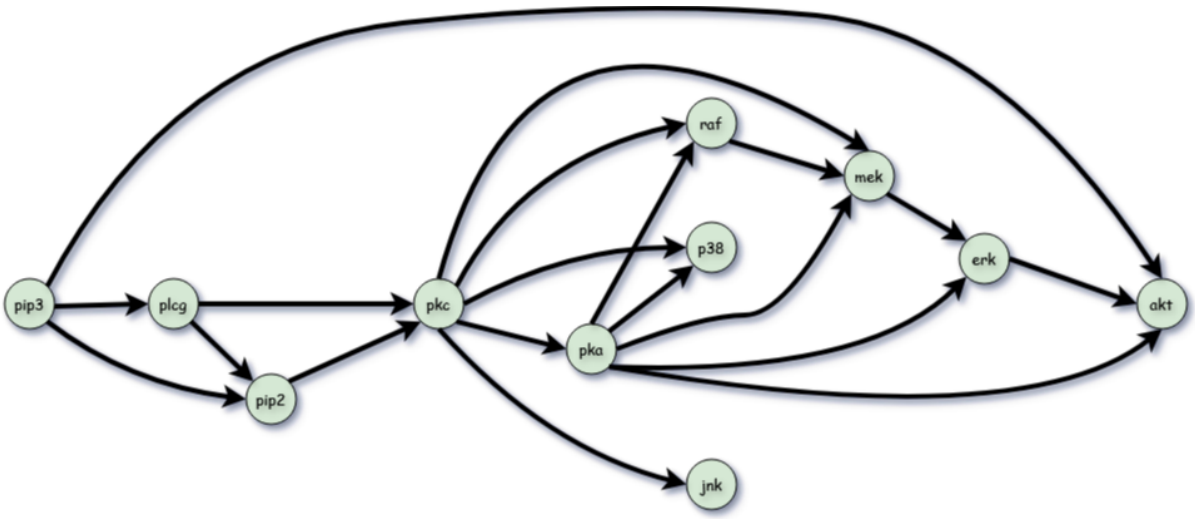


Figure 1.1: Gold-standard Raf pathway as described in [55]. The individual components of the network are various lipids, enzymes, and protein kinases [12]. The figure was created using Draw.io [22].

In the fields of biology and genomics, network modelling involves reconstructing networks such as the Raf pathway, or, more generally, any gene regulatory network [34, 50]. Extending the pioneering work done by Friedman *et al.* [21] and Hartemink *et al.* [28], substantial research on modelling gene regulatory networks from postgenomic data has been conducted by Werhli, Husmeier, and Grzegorczyk in [67, 66, 69, 68]. The modelling approach of the cited papers is based on Bayesian networks. Bayesian networks provide a flexible framework for representing probabilistic relationships on a graph: conditional dependencies and independencies are encoded directly in the graph structure. Specifically, the underlying architecture is a Directed Acyclic Graph (DAG). A family of conditional probability distributions is then associated with the DAG, so that each node can be interpreted as a random variable. We assume that each node x in the DAG is independent of any non-descendant node, given its parents (the nodes with directed edges pointing toward x). These components together define the Bayesian network [46]. Although Bayesian networks are traditionally used to model probabilistic dependencies among random variables, they admit a natural causal interpretation and, in special circumstances, give rise to causal networks [64, 47, 17]. Beyond applications in biology and genomics, Bayesian networks have been employed in engineering fault diagnosis [7], air-pollution modelling [65], financial portfolio risk analysis [56], and even military and defense scenarios [32, 18].

To aid clarity, we briefly summarize the structure of this paper. The paper is organized as a self-contained narrative: we first introduce the necessary mathematical preliminaries, then develop the theoretical underpinnings of our model, and conclude with simulation studies that showcase its practical performance.

In the first 4 sections of *Theoretical Framework*, we present the preliminary mathematical theory behind Bayesian networks, assuming an undergraduate-level understanding of basic statistics and

probability theory. This includes a discussion on Bayesianism and the principles of Bayesian inference, drawing primarily from [26]. We then present the foundational concepts of Bayesian networks, beginning with essential graph theory and probability theory before progressing to DAGs and Bayesian networks, with references to [35] and the Statistical Genomics (WMMA008-05) lecture notes by Pieter Trapman and Marco Grzegorczyk. A particular focus is given to DAG equivalence classes, especially CPDAGs, along with a brief introduction to Markov chain theory. Additionally, we present a scoring metric for Bayesian networks, assuming Gaussian data, following the approach in [27]. Readers already familiar with Bayesian networks may choose to skip these sections. We conclude *Theoretical Framework* by introducing the Bayesian model based on DAG coupling from [67] and describe in detail the MCMC sampling scheme used to approximate the posterior probability distribution for network inference.

The main original theoretical contributions of this thesis are presented in *Enhanced Model with New Coupling Schemes*, where we propose the CPDAG and skeleton coupling. Additionally, we develop an improved approximation algorithm that enhances CPU efficiency in computationally intensive simulations. We conclude by discussing a heuristic yet practical approach for integrating the new coupling schemes into the Bayesian model.

The creation of synthetic Raf data, selection of priors, input networks, and other parameters, as well as proposal distributions for the MCMC scheme, are described in *Experimental Setup*. We also outline and summarize the structure of our Python implementation used in the simulation studies.

Results of our simulations include an empirical assessment of the proposed approximation scheme, an evaluation of the impact and significance of the coupling mechanisms, an analysis of the model's robustness in inference, and further remarks on the convergence properties of the MCMC method. Although, in this paper, we specifically apply the Bayesian coupled model to synthetic data generated by the Raf signalling pathway, we emphasize that our model framework is inherently versatile: it can be used for arbitrary network modelling whenever data arise under multiple experimental conditions, and is therefore not restricted only to gene regulatory networks like Raf.

Finally, in *Discussion & Conclusion*, we suggest potential improvements for future research and critically assess the appropriateness of the methods proposed in this thesis.

2 Theoretical Framework

Sections 2.1–2.4 are optional for readers already familiar with Bayesian networks. Section 2.1 motivates Bayesian statistics and defines key terminology related to Bayes' theorem. Section 2.2 develops the formal definition of Bayesian networks. Section 2.3 explores the (Markov) equivalence of DAGs and shows how to characterize the DAG equivalence class using CPDAGs. Section 2.4 presents a scoring metric for DAGs and Bayesian networks: although it may appear tangential, it is essential for quantitatively distinguishing equivalent from non-equivalent structures and underpins the practical implementation of the models described later. Finally, Section 2.5 introduces the model from [67], to which we add an explicit analysis of its limiting behavior (see Equations 2.32 and 2.33).

While Sections 2.1–2.4 review foundational theory, drawing on standard references to ensure consistency, they also include additional theoretical remarks, new illustrative examples to clarify key concepts, and enhancements of existing literature examples. In particular, we work through a detailed, practical example to illustrate the notion of neighbor DAGs. We also include several network diagrams created in Draw.io; any figure without a citation is original to this work.

2.1 Few Notes about Bayesianism and Bayes Theorem

We begin by addressing some philosophical aspects of Bayesian inference before introducing Bayes' theorem. The earliest attempts to define probability in an abstract sense emerged from gambling, where understanding uncertainty was a crucial skill. Beyond gambling, probability plays a fundamental role in risk assessment, decision-making under uncertainty, and estimating the likelihood of rare events. For centuries, the nature of probability has been a topic of both mathematical and philosophical debate. The axioms of probability theory, as formulated by Kolmogorov in [36], quickly gained widespread acceptance. However, to this day, no universal consensus exists on the interpretation of probability, a debate that naturally extends also to statistical inference. The two dominant interpretations of probability are the frequentist ("objective") and Bayesian ("subjective") perspectives. The frequentist interpretation defines probabilities as long-run frequencies of events, while the Bayesian interpretation considers probabilities as subjective degrees of belief [26].

Consider a weather forecast stating that there is a 60% chance of rain tomorrow. Within the frequentist framework, this can be interpreted as: "On many previous days with the same atmospheric conditions (temperature, humidity, pressure, etc.), it rained approximately 60% of the time. The Law of Large Numbers applies." In contrast, the Bayesian perspective would suggest: "Given all available data (historical weather patterns, satellite images, forecasts, etc.), my degree of belief that it will rain tomorrow is 60%. If I were to place a bet on whether it will rain, I would consider 60% fair odds. If new information became available, such as satellite data indicating that a storm is moving away, I would update my belief accordingly."

Both philosophical interpretations of probability are certainly useful, and their applicability depends on the context. For instance, predicting the score of a specific student on a particular test does not align with the frequentist notion of probability. Notably, for smaller datasets, frequentist methods may be impractical, making Bayesian approaches the only viable option for statistical analysis. Bayesian methods have also recently gained prominence in Reinforcement Learning [25], important for fine-tuning stages of Large Language Models, such as the Generative Pre-trained Transformer (GPT) [52]. On the other hand, frequentist methods are often supported by more established programming packages, making them better suited for modelling scientific experiments or classical machine learning tasks.

In many cases, Bayesian and frequentist approaches represent two sides of the same coin. For example, what Bayesian statisticians refer to as priors, frequentist statisticians describe as regularization, among other conceptual parallels. According to the Bernstein–von Mises theorem, Bayesian and frequentist inferences are asymptotically equivalent under fairly weak conditions; however, this equivalence can break down in an infinite countable probability space [63].

Finally, it is worth noting that both frequentist and Bayesian interpretations are consistent with Kolmogorov's axioms. The differences between them arise only at the philosophical level, concerning how probability is interpreted.

From a Bayesian perspective, all unknown quantities, including both the parameter of interest and the data before observation, are assigned probability distributions. Assume we want to make an inference about the unknown parameter $\Theta = \theta$ by learning from data X. Based on X, we explore which values θ are probable and assess the level of uncertainty surrounding these estimates. We call the probability density function $f_{X|\Theta}(x \mid \theta)$ the likelihood. Bayesian inference also requires a prior distribution for Θ. The prior, $f_\Theta(\theta)$, may carry a varying degree of information based on our subjective belief or empirical evidence about the distribution of Θ. Alternatively, the prior can be chosen conventionally (for instance, as a uniform distribution), in which case we talk about an uninformative or an objective prior. The conditional probability density function $f_{\Theta|X}(\theta \mid x)$, called the posterior probability, can be computed using the Bayes theorem,

$$f_{\Theta|X}(\theta \mid x) = \frac{f_\Theta(\theta) f_{X|\Theta}(x \mid \theta)}{\int f_\Theta(\theta') f_{X|\Theta}(x \mid \theta') d\theta'}. \tag{2.1}$$

The denominator—marginal density of X is called the evidence and serves as a normalization constant. In the case of discrete random variables it becomes $\sum_{\theta' \in \Theta} f_\Theta(\theta') f_{X|\Theta}(x \mid \theta')$. The posterior quantifies our uncertainty in light of the data, in some sense upgrading $f_\Theta(\theta)$ to $f_{\Theta|X}(\theta \mid x)$. The evidence is often assumed to be constant, and in that case, we write

$$f_{\Theta|X}(\theta \mid x) \propto f_\Theta(\theta) f_{X|\Theta}(x \mid \theta). \tag{2.2}$$

2.2 DAGs and Bayesian Networks

We rely on basic knowledge of probability theory and statistics theory, as commonly taught in undergraduate programs. We begin by introducing the concept of conditional independence of random variables.

In a probability distribution $\mathbb{P}$, we say that a set of random variables X is conditionally independent of another set Y given a third set Z if, for every choice of values $x \in \mathrm{Val}(X)$, $y \in \mathrm{Val}(Y)$, and $z \in \mathrm{Val}(Z)$, we have

$$\mathbb{P}(X = x,\, Y = y \mid Z = z) \;=\; \mathbb{P}(X = x \mid Z = z)\, \mathbb{P}(Y = y \mid Z = z). \tag{2.3}$$

We write this relationship as $(X \perp Y \mid Z)$. The variables in Z are considered observed. When Z is empty, this reduces to the usual notion of marginal independence between X and Y. It is important to note that the independence of X and Y is not equivalent to the conditional independence of X and Y given Z. In fact, neither of these two definitions of independence implies the other.

Before defining probabilistic graphical models, namely the Bayesian networks, it is useful to first review fundamental terminology and concepts from graph theory. A graph $G = (\mathcal{X}, \mathcal{E})$, usually abbreviated as G, is a data structure consisting of a set of nodes $\mathcal{X}$ and edges $\mathcal{E}$. A pair of nodes X_i, X_j from the set of nodes $\mathcal{X} = \{X_1, \ldots, X_N\}$ can be connected by a directed edge, $X_i \to X_j$, or $X_i \leftarrow X_j$, or by undirected edge X_i—X_j. We say that the graph G is directed if all edges of G are directed. Similarly, we say that the graph G is undirected if all edges of G are undirected. The degree of a node X_i is the number of edges in which it participates. The degree of the graph G is the maximum over the degrees of every node in G. If X_i and X_j are connected by an edge of any type, we say that X_i and X_j are adjacent and denote this by $X_i \rightleftharpoons X_j$. If we have X_i—X_j, we say that X_i and X_j are neighbors. If we have $X_i \to X_j$, we say that X_i is a parent of X_j, while X_j is a child of X_i. The set of all parents of X_j in the graph G is denoted by $\mathrm{pa}_{X_j}[G]$.

In a graph G, a sequence of nodes $X_1, X_2, \ldots, X_k$ is called a path if each consecutive pair of nodes is connected by either a directed edge $X_i \to X_{i+1}$ or an undirected edge $X_i - X_{i+1}$. In other words, you can move from X_1 to X_k by following arrows or undirected links at each step. If, in addition, every link in that sequence is a directed arrow $X_i \to X_{i+1}$, then it is specifically called a directed path from X_1 to X_k. If there is a directed path from X_i to X_j in G, we say that X_i is an ancestor of X_j in G, while X_j is said to be the descendant of X_i in G. The converse statement also holds. Similar to the definition of a path, we might define a trial $X_1, \ldots, X_k$ in G if we have $X_i \rightleftharpoons X_{i+1}$ for every $i = 1, \ldots, k-1$. We say that a graph G is connected if for every X_i, X_j in G, there is a trial between X_i and X_j in G.

In a graph G, a cycle is a directed path $X_1, X_2, \ldots, X_k$ in which $X_1 = X_k$. A graph is called acyclic if it contains no such cycles. A cyclic path starting and ending at the same node X_i makes X_i its

own descendant. At the moment, we restrict ourselves to graphs that only contain directed edges and are acyclic. Acyclicity also excludes self-loops $X_i \to X_i$. These are called Directed Acyclic Graphs (DAGs). To distinguish between general graphs and DAGs, we write G and $\mathcal{G}$, respectively.

In a graph $G = (\mathcal{X}, \mathcal{E})$, a sequence of nodes $X_1, X_2, \ldots, X_N$ is called a topological ordering if, whenever there is a directed edge $X_i \to X_j$ in $\mathcal{E}$, then $i < j$. Note that G is a DAG if and only if it has a topological ordering. Also we note that given DAG $\mathcal{G}$ might have multiple topological orderings.

The skeleton of a DAG $\mathcal{G}$ is an undirected graph with the same number of nodes as $\mathcal{G}$ but with directed edges replaced by undirected edges. A v-structure or immorality is a substructure consisting of three nodes that look as follows: $X_i \to X_k \leftarrow X_j$. That is a set of three nodes where X_k is a child of two parents X_i, X_j such that X_i and X_j have no directed edge from one to the other.

Imagine a DAG $\mathcal{G}$ whose nodes $X_1, \ldots, X_N$ stand for random variables. This allows us to do probability on graphs. The edges in $\mathcal{G}$ capture dependencies between the random variables and, in some cases, may also represent direct causal relationships. When the edges indicate causality, the network is referred to as a causal network, a specific class of Bayesian networks [47]. Bayesian networks, first introduced by Judea Pearl in [46], are probabilistic graphical models that represent a set of random variables and their conditional dependencies on a DAG. Each node is associated with a probability distribution that defines the chance of the node being in a particular state. Conditional probabilities are used when the state of one node depends on the state of another. These dependencies propagate throughout the network, influencing the probabilities of other nodes, which are updated as new information becomes available [30]. For example, the nodes can represent biological molecules or genes [50].

To formally define Bayesian networks, we introduce the local Markov assumption. This assumption states that, within a given Bayesian network, each random variable X_i from the underlying DAG $\mathcal{G}$ is conditionally independent of its non-descendants, denoted as $\text{NonDescendants}_{X_i}$, given its parents in $\mathcal{G}$ $\text{pa}_{X_i}[\mathcal{G}]$. That is, $(X_i \perp \text{NonDescendants}_{X_i} \mid \text{pa}_{X_i}[\mathcal{G}])$.

The local Markov assumption plays a fundamental role in the structure of Bayesian networks, ensuring that dependencies are only specified between connected nodes and that inference can be carried out efficiently. In causal settings, this aligns with the philosophical idea that causes directly determine their effects. If we intervene on some node, its parents are the direct influencers, and other ancestors do not have a direct effect beyond what is mediated through the parents. Next, we demonstrate that, in a given Bayesian network, the local Markov assumption leads to a factorization of its joint probability distribution, encoding conditional independence relationships between the variables. It is useful to first define a set of independencies associated with a distribution $\mathbb{P}$. We also introduce the formal notion of I-maps.

For a probability distribution $\mathbb{P}$ defined over the variables $\mathcal{X} = (X_1, \ldots, X_N)$, let $\mathcal{I}(\mathbb{P})$ denote the collection of all conditional independence statements of the form $(X \perp Y \mid Z)$ that hold in $\mathbb{P}$. Similarly, for any graph G, let $\mathcal{I}(G)$ be the set of independence assertions encoded by G. We say that G is an I-map for a given set of independencies $\mathcal{I}$ if $\mathcal{I}(G) \subset \mathcal{I}$. A DAG $\mathcal{G}$ is an I-map for $\mathbb{P}$ if $\mathcal{G}$ is an I-map for $\mathcal{I}(\mathbb{P})$.

A distribution $\mathbb{P}$ over the variables $X_1, \ldots, X_N$ is said to factorize over a DAG $\mathcal{G}$ if its joint probability can be written as the product of local conditionals for each node given its parents in $\mathcal{G}$. That is, if $\text{pa}_{X_i}[\mathcal{G}]$ denotes the set of parents of X_i in the graph $\mathcal{G}$, then

$$\mathbb{P}(X_1, \ldots, X_N) \;=\; \prod_{i=1}^{N} \mathbb{P}\big(X_i \mid \text{pa}_{X_i}[\mathcal{G}]\big). \tag{2.4}$$

This decomposition is often referred to as the chain rule for Bayesian networks, because it expresses the full joint distribution as a sequence of conditional probabilities aligned with the DAG structure. Bayesian networks can be defined as a pair $(\mathcal{G}, \mathbb{P})$ where $\mathbb{P}$ factorizes over $\mathcal{G}$, and where $\mathbb{P}$ is specified as a set of conditional probability densities associated with the nodes of $\mathcal{G}$.

Theorem 2.1 *Let $\mathcal{G}$ be a Bayesian network structure over a set of random variables $\mathcal{X} = (X_1, \ldots, X_N)$, and let $\mathbb{P}$ be a joint distribution over the same space. If $\mathcal{G}$ is an I-map for $\mathbb{P}$, then $\mathbb{P}$ factorizes according to $\mathcal{G}$.*

Proof: As $\mathcal{G}$ is a Bayesian network structure, it is a DAG. It is, therefore possible to construct a topological ordering $X_1, \ldots, X_N$ relative to $\mathcal{G}$. For any joint distribution, it holds that

$$\mathbb{P}(X_1, \ldots, X_N) = \prod_{i=1}^{N} \mathbb{P}(X_i \mid X_1, \ldots, X_{i-1}).$$

Consider one of the factors $\mathbb{P}(X_i \mid X_1, \ldots, X_{i-1})$. As $\mathcal{G}$ is an I-map for $\mathbb{P}$, in particular the local Markov assumption

$$(X_i \perp \text{NonDescendants}_{X_i} \mid \text{pa}_{X_i}) \in \mathcal{I}(\mathbb{P}).$$

As we consider the topological ordering, all the parents of X_i are in the set $X_1, \ldots, X_{i-1}$, and none of X_i's descendants can be in the set. Therefore,

$$\{X_1, \ldots, X_{i-1}\} = \text{pa}_{X_i} \cup Z,$$

where $Z \subseteq \text{NonDescendants}_{X_i}$. Using the local Markov assumption, we obtain

$$\begin{aligned}\mathbb{P}(X_i \mid X_1, \ldots, X_{i-1}) &= \frac{\mathbb{P}((Z \cup X_i) \setminus \text{pa}_{X_i} \mid \text{pa}_{X_i})}{\mathbb{P}(Z \setminus \text{pa}_{X_i} \mid \text{pa}_{X_i})} = \frac{\mathbb{P}((Z \setminus \text{pa}_{X_i}) \cup (X_i \setminus \text{pa}_{X_i}) \mid \text{pa}_{X_i})}{\mathbb{P}(Z \setminus \text{pa}_{X_i} \mid \text{pa}_{X_i})} \\ &= \frac{\mathbb{P}(X_i \mid \text{pa}_{X_i})\mathbb{P}(Z \setminus \text{pa}_{X_i} \mid \text{pa}_{X_i})}{\mathbb{P}(Z \setminus \text{pa}_{X_i} \mid \text{pa}_{X_i})} = \mathbb{P}(X_i \mid \text{pa}_{X_i}).\end{aligned}$$

The result follows by applying this transformation to every term in the factorization. □

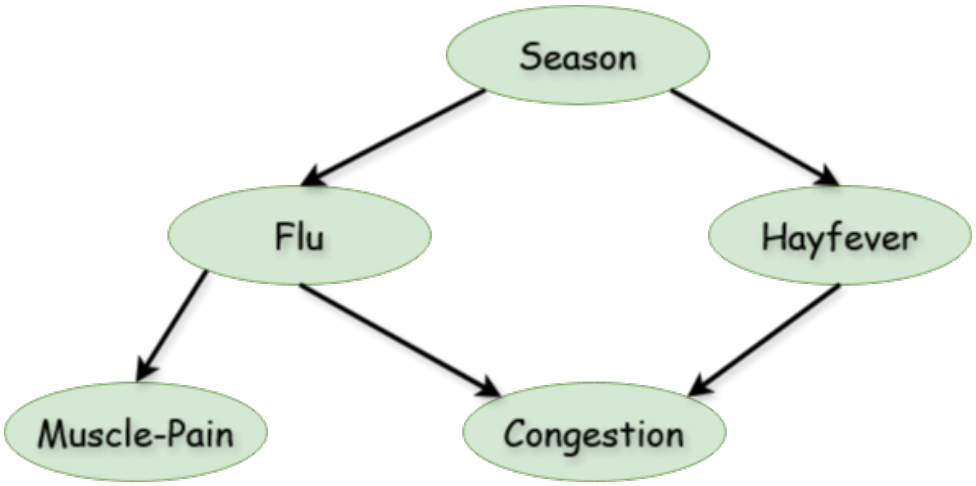


Figure 2.1: Sample Bayesian network from [35]. The figure was created using Draw.io [22].

For example, consider a sample Bayesian network as depicted in Figure 2.1. Let us denote the variable names with their initial capital letters. It is possible to construct 4 different topological orderings:

1. (S, F, H, C, M);
2. (S, F, H, M, C);
3. (S, H, F, C, M);
4. (S, H, F, M, C).

In addition, the local Markov assumption yields various independencies such as:

1. $(F \perp H \mid S)$;
2. $(C \perp S \mid F, H)$;
3. $(M \perp H, C \mid F)$;
4. $(M \perp C \mid F)$; ...

The factorization reads

$$\mathbb{P}(S, F, H, M, C) = \mathbb{P}(S)\mathbb{P}(F \mid S)\mathbb{P}(H \mid S)\mathbb{P}(M \mid F)\mathbb{P}(C \mid F, H).$$

2.3 Representation and Equivalence Classes of DAGs

It is more practical to view a DAG $\mathcal{G}$ on N nodes $X_1, \ldots, X_N$ through its $N \times N$ adjacency matrix[†] defined as $\mathcal{A}_{ij} = \mathbb{1}\{X_i \to X_j\}$. That is, $\mathcal{A}_{ij} = 1$ if there is a directed edge pointing from X_i to X_j, and $\mathcal{A}_{ij} = 0$ if there is no edge connecting X_i and X_j, for some $i \neq j : i, j \in \{1, \ldots, N\}$. The non-zero entries of the matrix $E = \sum_{k=1}^{N} \mathcal{A}^k$ encode the ancestors of $\mathcal{G}$. We call the matrix A defined as $\mathbb{1}\{A_{ij} = 1\} = \mathbb{1}\{E_{ij} > 0\}$ the ancestor matrix. We have $A_{ij} = \mathbb{1}\{X_i \text{ is an ancestor of } X_j\}$, for $i, j = 1, \ldots, N$. To illustrate this, consider the DAG 2.1 and denote $X_1 \coloneqq S, X_2 \coloneqq F, X_3 \coloneqq H, X_4 \coloneqq M, X_5 \coloneqq C$. The adjacency matrix of the DAG is

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.5}$$

and the ancestor matrix is

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.6}$$

Next, we introduce the concept of neighboring DAGs. Given a DAG $\mathcal{G}$, we define another DAG $\mathcal{G}' \neq \mathcal{G}$ as a neighbor of $\mathcal{G}$ if $\mathcal{G}'$ can be obtained by applying one of the following single-edge operations:

1. Deletion of a directed edge in $\mathcal{G}$;
2. Addition of a directed edge to $\mathcal{G}$;
3. Reversal of a directed edge in $\mathcal{G}$.

We denote the set of all neighboring DAGs to $\mathcal{G}$ by $\mathcal{N}(\mathcal{G})$. We can determine the cardinality of $\mathcal{N}(\mathcal{G})$ by counting the number of single-edge operations that are possible in $\mathcal{G}$.

Notice that deleting a directed edge in a given DAG $\mathcal{G}$ is never a problem, because the resulting DAG $\mathcal{G}'$ always satisfies the acyclicity criterion. However, adding or reversing a directed edge may introduce a cycle. Since reversing an edge can be viewed as first deleting the edge and then adding one in the opposite direction, it suffices to analyze the addition operation. For instance, adding the edge $C \to S$ in the DAG 2.1 results in a cycle. In general, adding the edge from X_i to X_j is possible if and only if X_j is not an ancestor of X_i, which is equivalent to requiring $A_{ji} \neq 1$. Assume that we have a DAG $\mathcal{G}$ with an adjacency matrix $\mathcal{A}$. The allowed additions in $\mathcal{G}$ can be obtained as follows. Start with a matrix full of ones $\mathbf{1}_{N \times N}$. Since we do not allow self-loops in DAGs, we can exclude the diagonal terms of $\mathbf{1}_{N \times N}$. We can also exclude the entries $\mathbb{1}\{A_{ij} = 1\}$ since we cannot duplicate already existing edges. Lastly, to prevent introducing cycles, we exclude the entries $\mathbb{1}\{A_{ij}^T = 1\}$. Thus, the non-zero entries of the matrix $\mathbf{1}_{N \times N} - \mathbf{Id}_{N \times N} - \mathcal{A} - A^T$ represent the edges that can be added without introducing a cycle. In our example this is

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.7}$$

meaning that, to create a neighbor of $\mathcal{G}$, we can add one of the edges: $S \to M, S \to C, F \to H, H \to F, H \to M, M \to H, M \to C, C \to M$.

Next, we study the equivalence classes of Bayesian networks by their associated DAGs. Consider for instance the DAG $\mathcal{G}$: $X_1 \to X_2 \to X_3$, and the DAG $\mathcal{G}'$: $X_1 \leftarrow X_2 \to X_3$. Let us study the factorization $\mathbb{P}(X_1, \ldots, X_N) = \prod_{i=1}^{N} \mathbb{P}(X_i \mid \text{pa}_{X_i})$ for each of these DAGs. We have

[†]Depending on the literature source, adjacency matrices are also called incidence matrices.

- The DAG $\mathcal{G}$ has a factorization

$$\mathbb{P}(X_1, X_2, X_3) = \mathbb{P}(X_1)\mathbb{P}(X_2 \mid X_1)\mathbb{P}(X_3 \mid X_2) = \frac{\mathbb{P}(X_1)\mathbb{P}(X_1, X_2)\mathbb{P}(X_2, X_3)}{\mathbb{P}(X_1)\mathbb{P}(X_2)} \tag{2.8}$$

$$= \frac{\mathbb{P}(X_1, X_2)\mathbb{P}(X_2, X_3)}{\mathbb{P}(X_2)}. \tag{2.9}$$

- The DAG $\mathcal{G}'$ has a factorization

$$\mathbb{P}(X_1, X_2, X_3) = \mathbb{P}(X_1 \mid X_2)\mathbb{P}(X_2)\mathbb{P}(X_3 \mid X_2) = \frac{\mathbb{P}(X_1, X_2)\mathbb{P}(X_2)\mathbb{P}(X_2, X_3)}{(\mathbb{P}(X_2))^2} \tag{2.10}$$

$$= \frac{\mathbb{P}(X_1, X_2)\mathbb{P}(X_2, X_3)}{\mathbb{P}(X_2)}. \tag{2.11}$$

We observe that both DAGs yield the same factorization. This follows from the fact that $\mathcal{G}$ and $\mathcal{G}'$ encode the same set of independence relations.

Two DAGs $\mathcal{G}$ and $\mathcal{G}'$ defined over the same set of variables $\mathcal{X} = (X_1, \ldots, X_N)$ are said to be $\mathcal{I}$-equivalent if they encode exactly the same conditional independencies $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{G}')$. As a result, the set of all DAGs over $\mathcal{X}$ is partitioned into a set of mutually exclusive and exhaustive $\mathcal{I}$-equivalence classes, which are the set of equivalence classes induced by the $\mathcal{I}$-equivalence relation. For brevity, we use the term equivalent and denote this by $\mathcal{G} \equiv \mathcal{G}'$. To determine whether two DAGs (or Bayesian networks) are equivalent, rather than directly performing the factorization 2.4, we can analyze their skeletons and v-structures. Two DAGs are equivalent if and only if they share the same skeleton and the same number of v-structures [64]. In what follows, we present a method to obtain the full equivalence class of a DAG $\mathcal{G}$.

Given some DAG, if we extract its skeleton and assign the direction to the edges participating in the v-structures, we obtain a graph that contains both directed and undirected edges. Such graphs are called Partially Directed Acyclic Graphs (PDAG). However, the description of the equivalence class using only the skeletons and v-structures is somewhat incomplete. For instance, if we know that our DAG is in a certain equivalence class where two nodes X and Y are connected by an edge, we do not necessarily know the direction of that edge. Is it $X \to Y$ or $X \leftarrow Y$? We call such edges reversible. If, on the other hand, the direction of the edge is fully determined, for instance, by the v-structures, we call the edge compelled. Even though not every dependency in the DAG (and its equivalence class) can be learned by data during an inference phase, we can further extend PDAGs into Completed Partially Directed Acyclic Graphs (CPDAG). A PDAG $\tilde{\mathcal{G}}$ is a CPDAG of the equivalence class of $\mathcal{G}$ if it shares the same skeleton as $\mathcal{G}$, and contains a directed edge $X \to Y$ if and only if all DAGs $\mathcal{G}' : \mathcal{G}' \equiv \mathcal{G}$ contain the edge $X \to Y$. CPDAGs fully characterize the partitioning of DAGs into equivalence classes. If an edge is directed, all members of the equivalence class agree on its orientation[†].

It is clear that edges participating in the v-structures of G are compelled in $\tilde{\mathcal{G}}$. Additional edges in $\tilde{\mathcal{G}}$ can become compelled if the newly assigned directions (a) do not introduce a cycle, and (b) do not create additional v-structure. We have 3 rules for such new assignments. These rules are shown in Figure 2.2.

In summary, the DAG-CPDAG algorithm [10] works as follows.

1. Given a DAG $\mathcal{G}$, construct its skeleton $\mathcal{G}_0$;
2. For every edge that participates in a v-structure in $\mathcal{G}$, replace the corresponding undirected edge in $\mathcal{G}_0$ with a directed edge. This yields a PDAG $\mathcal{G}_1$;
3. For a PDAG $\mathcal{G}_i$, obtain $\mathcal{G}_{i+1}$ by applying any of the rules shown in Figure 2.2;
4. Repeat 3. until no rule from Figure 2.2 can be applied.

Illustrative example of the DAG-CPDAG algorithm can be seen in Figure 2.3.

[†]However, if an edge is undirected, different DAGs within the equivalence class may assign opposite orientations to it. This limitation of the Bayesian network approach is known as statistical indistinguishability [35].

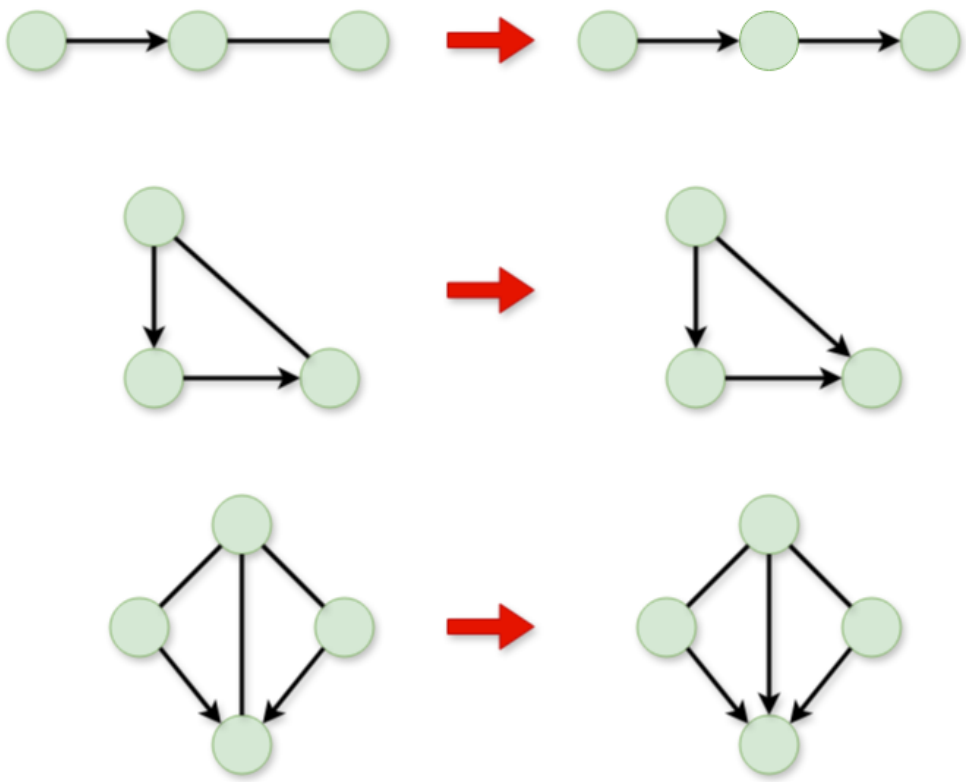


Figure 2.2: Rules for assigning directions in PDAGs. Once a skeleton and v-structures are identified, we examine all subgraphs and decide whether any of the rules displayed on the figure can be applied. When this process is finished, we obtain the CPDAG. The figure was created using Draw.io [22].

We conclude this section with a brief introduction of the theory of Markov chains. A Markov chain is a stochastic process $(X_0, X_1, \dots)$ in which if we are at a point $X_t = x$, we move to the next point $X_{t+1} = y$ with a fixed probability $T(x, y)$ that depends only on x. Formally, if we denote $H_t = \cap_{s=0}^{t-1}\{X_s = x_s\}$, then, assuming that $\mathbb{P}(H_{t-1} \cap \{X_t = x_t\}) > 0$, we have

$$\mathbb{P}(X_{t+1} = y \mid H_{t-1} \cap \{X_t = x\}) = \mathbb{P}(X_{t+1} = y \mid X_t = x) = T(x, y) \tag{2.12}$$

for all $t \geq 1$ and for all x, y from the state space S. The matrix $\mathbf{T} = T(x, y), \forall x, y \in S$ is called the transition kernel matrix. Notice that 2.12 resembles the local Markov property from Section 2.2. The $|S| \times |S|$ matrix $\mathbf{T}$ is stochastic, which means that

$$\sum_{y \in S} T(x, y) = 1, \text{ for all } x \in S. \tag{2.13}$$

The Markov chain is called finite if the state space S is of finite size, and throughout this paper, we work with finite Markov chains. In general, the distribution at time $t+1$ can be found by matrix multiplication:

$$\mathbb{P}(X_{t+1} = y) = \mathbb{P}_{t+1}(y) = \sum_{x \in S} \mathbb{P}(X_t = x_t) T(x, y) = \sum_{x \in S} \mathbb{P}_t(x) T(s, y), \text{ for all } y \in S. \tag{2.14}$$

Consider a discrete finite Markov chain with a transition kernel matrix $\mathbf{T}$ and state space $S = \{x_0, \dots, x_k\}$ and define $p(x_j)$ to be the probability of $x_j \in S$. Set $\pi = (p(x_0), \dots, p(x_k))$. The probabilistic distribution π is called stationary if we have

$$\pi = \pi \mathbf{T}. \tag{2.15}$$

Markov chain is called irreducible if for every pair $x, y \in S$ there exists an integer t such that $T^t(x, y) > 0$. Denote by $\mathcal{T}(x) = \{t \geq 1 : T^t(x, x) > 0\}$ the set of times $t \geq 1$ such that if we start at the state x we return to x after $t \geq 1$ iterations. The greatest common divisor of $\mathcal{T}(x)$ is called the period of state x. If every state in the chain has a period equal to 1, we call such chain aperiodic. Markov chains that are both irreducible and aperiodic are called ergodic.

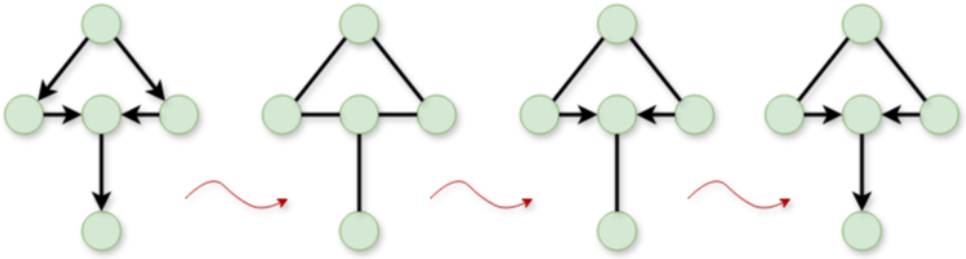


Figure 2.3: From the left, the first network is the given DAG. Next, a skeleton is extracted, and v-structures are identified. Lastly, we apply the top rule from Figure 2.2 and obtain the CPDAG. The figure was created using Draw.io [22].

For proofs of the following claims, we refer to the book by Levin et al. [38]. It can be shown that any ergodic and discrete Markov chain has a unique stationary distribution π and that the chain converges in distribution to π regardless of the initial distribution $p(x_0)$. In fact, a stronger result can be proven: convergence in total variation distance. To define this rigorously, let us assume we have two probability distributions ϑ and μ on a state space S. The total variation distance between these two distributions is defined as:

$$\|\vartheta - \mu\|_{\mathrm{TV}} = \max_{A \subseteq S} |\vartheta(A) - \mu(A)|. \tag{2.16}$$

Assume we have a discrete Markov chain that is ergodic. Assume the chain is associated with a transition kernel matrix $\mathbf{T}$, state space S, and a stationary distribution π. Then there exist constants $\alpha \in (0,1)$ and $C > 0$ such that

$$\max_{x \in S} \|T^t(x, \cdot) - \pi\|_{\mathrm{TV}} \leq C\alpha^t. \tag{2.17}$$

This is known as the Markov chain convergence theorem.

2.4 BGe Score

Assume we have a DAG $\mathcal{G}$ on N nodes $X_1, \ldots, X_N$ and let $\mathcal{D}$ be a $N \times m$ matrix. The matrix $\mathcal{D}$ in this setting represents the observed data, and its columns represent the independent m realizations of the nodes $X_1, \ldots, X_N$. The Bayes theorem for computing the posterior probability $\mathbb{P}(\mathcal{G} \mid \mathcal{D})$ tells us that

$$\mathbb{P}(\mathcal{G} \mid \mathcal{D}) = \frac{\mathbb{P}(\mathcal{D} \mid \mathcal{G})\mathbb{P}(\mathcal{G})}{\sum_{\mathcal{G}^* \in \Omega} \mathbb{P}(\mathcal{D} \mid \mathcal{G}^*)\mathbb{P}(\mathcal{G}^*)}, \tag{2.18}$$

where Ω denotes the space of all possible DAGs on N nodes, $\mathbb{P}(\mathcal{G})$ and $\mathbb{P}(\mathcal{G}^*)$ denote the prior distribution for the DAGs. In this section, we are interested in the likelihood term $\mathbb{P}(\mathcal{D} \mid \mathcal{G})$. Assume every node X_i has an associated parameter vector q_i with it. From the law of total probability, we obtain the following,

$$\mathbb{P}(\mathcal{D} \mid \mathcal{G}) = \int \mathbb{P}(\mathcal{D}, q \mid \mathcal{G}) dq = \int \mathbb{P}(\mathcal{D} \mid \mathcal{G}, q)\mathbb{P}(q \mid \mathcal{G}) dq. \tag{2.19}$$

If we fix q and consider the Bayesian network with $\mathcal{G}$ as its underlying DAG, we can factorize $\mathbb{P}(\mathcal{D} \mid \mathcal{G}, q)$ using Eq. 2.4,

$$\mathbb{P}(\mathcal{D} \mid \mathcal{G}, q) = \prod_{i=1}^{N} \prod_{j=1}^{m} \mathbb{P}(X_i = \mathcal{D}_{ij} \mid \mathrm{pa}_{X_i}[\mathcal{G}] = \mathcal{D}_{\mathrm{pa}_{X_i}[\mathcal{G}]j}, q_i). \tag{2.20}$$

We assume that the probability of each parameter q_i depends only on the parent configuration of the node X_i, that is, $\mathbb{P}(q_i \mid \mathcal{G}) = \mathbb{P}(q_i \mid \mathrm{pa}_{X_i}[\mathcal{G}])$ and we also assume overall parameter independence in a sense that

$$\mathbb{P}(q \mid \mathcal{G}) = \prod_{i=1}^{N} \mathbb{P}(q_i \mid \mathrm{pa}_{X_i}[\mathcal{G}]). \tag{2.21}$$

If we substitute 2.20 and 2.21 into 2.19, we obtain

$$\begin{aligned}\mathbb{P}(\mathcal{D} \mid \mathcal{G}) &= \int \prod_{i=1}^{N} \prod_{j=1}^{m} \mathbb{P}(X_i = \mathcal{D}_{ij} \mid \mathrm{pa}_{X_i}[\mathcal{G}] = \mathcal{D}_{\mathrm{pa}_{X_i}[\mathcal{G}]j}, q_i) \prod_{i=1}^{N} \mathbb{P}(q_i \mid \mathrm{pa}_{X_i}[\mathcal{G}]) dq \\ &= \prod_{i=1}^{N} \Big(\int \mathbb{P}(q_i \mid \mathrm{pa}_{X_i}[\mathcal{G}]) \Big(\prod_{j=1}^{m} \mathbb{P}(X_i = \mathcal{D}_{ij} \mid \mathrm{pa}_{X_i}[\mathcal{G}] = \mathcal{D}_{\mathrm{pa}_{X_i}[\mathcal{G}]j}, q_i) \Big) dq_i \Big).\end{aligned} \tag{2.22}$$

The proof of the last inequality can be found in [23]. Denote

$$\Psi[\mathcal{D}_i^{\mathrm{pa}_{X_i}[\mathcal{G}]}] = \Big(\int \mathbb{P}(q_i \mid \mathrm{pa}_{X_i}[\mathcal{G}]) \Big(\prod_{j=1}^{m} \mathbb{P}(X_i = \mathcal{D}_{ij} \mid \mathrm{pa}_{X_i}[\mathcal{G}] = \mathcal{D}_{\mathrm{pa}_{X_i}[\mathcal{G}]j}, q_i) \Big) dq_i \Big), \tag{2.23}$$

where $\mathcal{D}_i^{\mathrm{pa}_{X_i}[\mathcal{G}]} = \{(\mathcal{D}_{ij}, \mathcal{D}_{\mathrm{pa}_{X_i}[\mathcal{G}]j}) : 1 \leq j \leq m\}$. The local scores $\Psi[\mathcal{D}_i^{\mathrm{pa}_{X_i}[\mathcal{G}]}]$ can be computed based on the stochastic model for $\mathcal{D}_i^{\mathrm{pa}_{X_i}[\mathcal{G}]}$. In our case, we use a linear Gaussian model with a normal-Wishart prior - the so-called BGe scoring metric. This metric for learning Gaussian Bayesian networks was first introduced in [23]. We assume that each of the m data observations $\mathcal{D}_{\cdot,j}$ (the rows of the matrix $\mathcal{D}$) are taken from a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, with unknown mean μ and unknown covariance matrix Σ. We call $W = \Sigma^{-1}$ the precision matrix. The distribution of μ given W is $\mathcal{N}(\mu_0, (\nu W)^{-1})$, $\nu > 0$. W follows a Wishart distribution $W(\alpha, T_0)$, with $\alpha > N + 1$ degrees of freedom and a covariance matrix T_0. Assuming that W is a positive definite $N \times N$ matrix, the density of the Wishart distribution is given by

$$f_W(W) = \frac{|W|^{\frac{\alpha - N - 1}{2}} e^{-\mathrm{Trace}(T_0^{-1} W)}}{2^{\frac{\alpha N}{2}} \pi^{\frac{N(N-1)}{4}} \prod_{j=1}^{N} \Big(\Gamma\Big(\frac{\alpha}{2} - \frac{j-1}{2} \Big) \Big) |T_0|^{\frac{\alpha}{2}}}, \tag{2.24}$$

where $\Gamma(\cdot)$ is the gamma function $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$ with $Re(z) > 0$, $\mathrm{Trace}(A) = \sum_{i=1}^{N} a_{ii}$ is the trace of an $N \times N$ matrix A, and $|\cdot|$ denotes the determinant. The Wishart distribution is a conjugate prior for the Gaussian distribution. Next, define

$$T_{\mathcal{D},m} = T_0 + S_{\mathcal{D},m} + \frac{\nu m}{\nu + m} (\mu_0 - \bar{\mathcal{D}})(\mu_0 - \bar{\mathcal{D}})^T, \tag{2.25}$$

where $\bar{\mathcal{D}}$ is the mean value of the m-th row of $\mathcal{D}$ and

$$S_{\mathcal{D},m} = \sum_{j=1}^{m} (\mathcal{D}_{\cdot,j} - \bar{\mathcal{D}})(\mathcal{D}_{\cdot,j} - \bar{\mathcal{D}})^T \tag{2.26}$$

The prior joint distribution for μ and $W = \Sigma^{-1}$ is thus the normal-Wishart distribution. The choice of hyperparameters μ_0 and T_0 can reflect our prior knowledge about the mean (in principle unknown) μ and the covariance matrix Σ of the normal distribution from which the data observations $\mathcal{D}_{\cdot,j}$ are assumed to be sampled. The choice $\alpha > N + 1$ and $\nu = 1$ ensures a weakly informative prior.

The likelihood of any $\mathcal{D}^S \subseteq \mathcal{D}$ consisting of m realizations of the K-dimensional subset $S \subseteq \{X_1, \ldots, X_N\}$ is given by

$$\mathbb{P}(\mathcal{D}^S \mid \mathcal{G}_C(S)) = (2\pi)^{-\frac{Km}{2}} \Big(\frac{\nu}{\nu + m} \Big)^{\frac{K}{2}} \frac{c(K, \alpha)}{c(K, \alpha + m)} |T_0^S|^{\frac{\alpha}{2}} |T_{\mathcal{D},m}^S|^{-\frac{\alpha + m}{2}}, \tag{2.27}$$

where $\mathcal{G}_C$ denotes the complete DAG with maximum number of edges, T_0^S and $T_{\mathcal{D},m}^S$ consist of those K rows and columns of T_0 and $T_{\mathcal{D},m}$ that correspond to variables in S, and $c(x, y)$ is defined as

$$c(x, y) = \Big(2^{\frac{xy}{2}} \pi^{\frac{x(x-1)}{4}} \prod_{i=1}^{x} \Gamma\Big(\frac{y + 1 - i}{2} \Big) \Big)^{-1}. \tag{2.28}$$

Every scoring network for Bayesian networks has to satisfy the score equivalence. That means that we expect identical scores for equivalent networks. Under an assumption of Gaussian data, Geiger and Heckerman show in [23] that there are only two scoring metrics satisfying the score equivalence. For continuous data, this is the just-described BGe score. For discrete data, this is the multinomial distribution with a Dirichlet prior - the so-called BDe score.

2.5 The Bayesian Coupling Scheme: A Starting Point

One of the main objectives of this paper is to reconstruct the Raf pathway, as shown in Figure 1.1, using synthetic Raf data. However, research has shown [69] that gene regulatory networks, such as the Raf pathway, are not fixed entities. In the context of inferring gene regulatory networks associated with an organism's immune system, some subpathways may only activate in response to specific conditions, such as infection, and remain undetectable if gene expression profiles are only collected under healthy conditions. For example, when inferring the structure of Raf using data from patient A, some Raf subpathways that are inactive in patient A may be active in patient B, leading to differences in the inferred structure. However, there are often considerable commonalities, as some subpathways may remain identical across patients. Overall, the activation or inactivation of certain Raf subpathways may vary depending on experimental and environmental conditions [67].

Consider two extreme strategies for reconstructing the Raf pathway. First, we could ignore fluctuations in experimental conditions and merge all data into a single, uniform dataset. While this approach might simplify the problem, it blurs the distinctions between the network states in healthy and infected cells. Second, we could keep the data from different conditions separate and reconstruct independent regulatory subnetworks active under each condition. However, this strategy has significant drawbacks. It inevitably leads to information loss, which in turn reduces statistical power and accuracy. Postgenomic datasets are often sparse and not particularly extensive, so the ability to accurately reconstruct networks is already compromised. Dividing these already limited datasets into smaller subsets only increases uncertainty, making it even more challenging to infer reliable network structures [69, 3]. Furthermore, there is a conceptual issue with this strategy, as it assumes the subnetworks are independent. Although Raf may shift between different active states in response to various external signals, current scientific consensus suggests that it most likely shares a common underlying architecture [55]—the gold-standard pathway depicted in Figure 1.1.

A coupling-based[†] compromise between the two extreme strategies discussed earlier, facilitating information sharing between distinct datasets, was proposed in [67]. The approach introduces a probabilistic model in which separate subnetworks are learned from different gene expression datasets, but these subnetworks are interconnected by loosely enforcing their similarity to a common underlying generic network.

Assume we have I datasets $\mathcal{D}_1, \ldots, \mathcal{D}_I$, each obtained under potentially different experimental condition. Details on how we use the gold-standard Raf pathway to generate synthetic datasets are provided in Section 4.1. Next, each of the I datasets correspond to a one of the network structures $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(I)}$. We associate the individual network structures with the hyperparameters $\beta^{(1)}, \ldots, \beta^{(I)}$. Lastly, we tie each $\mathcal{G}^{(i)}$, for $i = 1, \ldots, I$, to a new network $\mathcal{G}^*$ called the hypernetwork. This induces a direct coupling between the network structures $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(I)}$ and the hypernetwork $\mathcal{G}^*$ and an indirect coupling between the network structures themselves. The hypernetwork encourages the individual networks to be similar. This Bayesian probabilistic graphical model is displayed in Figure 2.4. In our work, we assume that each network structure in the model is a valid DAG. This is different from the model from [67], where the authors do not require the hypernetwork $\mathcal{G}^*$ to be acyclic.

From Theorem 2.1, we know that the joint density of the Bayesian probabilistic graphical model 2.4 factorizes as

$$\mathbb{P}(\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(I)}, \mathcal{D}_1, \ldots, \mathcal{D}_I, \beta^{(1)}, \ldots, \beta^{(i)}, \mathcal{G}^*) = \prod_{i=1}^{I} \mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)}) \mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}, \mathcal{G}^*) \mathbb{P}(\beta^{(i)}) \mathbb{P}(\mathcal{G}^*). \quad (2.29)$$

The prior distribution over the network structures is taken to be the Gibbs distribution[†],

[†]Coupling is a powerful method in probability theory that provides a framework for comparing random variables by constructing a joint distribution for them, which allows us to study their behavior together in a way that may not be directly possible by analyzing them individually. Assume we have two probability distributions ϑ, μ defined on Ω. The coupling between ϑ μ is a function on $\Omega \times \Omega$ by considering a pair of variables (X, Y) satisfying the following: $\sum_{y \in \Omega} \mathbb{P}(X = x, Y = y) = \mu(x)$ and $\sum_{x \in \Omega} \mathbb{P}(X = x, Y = y) = \vartheta(y)$. That is, μ is the marginal distribution of X and ϑ is a marginal distribution of Y [38].

[†]Depending on the literature, Gibbs distribution is also known as Boltzmann distribution.

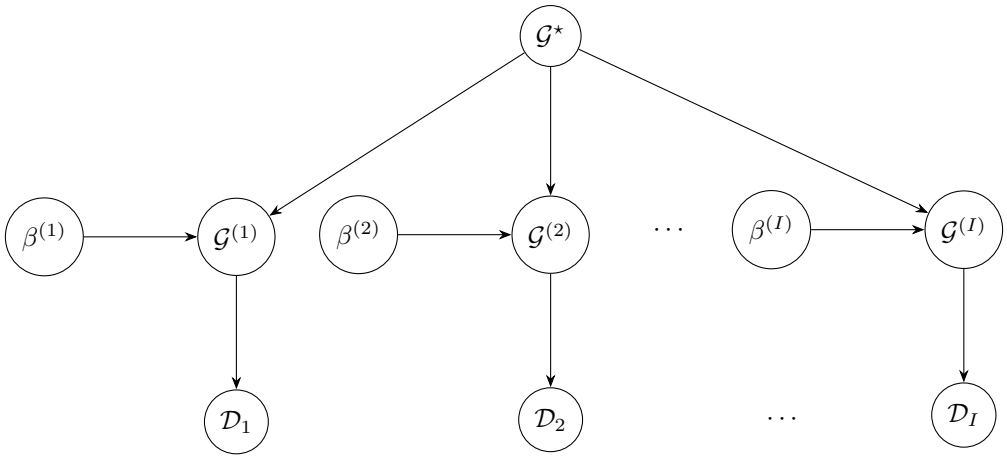


Figure 2.4: The probabilistic model for learning several active subnetworks. The hypernetwork $\mathcal{G}^*$ leads to a coupling between the individual subnetworks, forcing them to be similar.

$$\mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}, \mathcal{G}^*) = \frac{e^{-\beta^{(i)}\|\mathcal{G}^{(i)} - \mathcal{G}^*\|}}{Z(\beta^{(i)}, \mathcal{G}^*)}, \text{ for } i = 1, \ldots, I, \tag{2.30}$$

where $\|\mathcal{G}^{(i)} - \mathcal{G}^*\|$ measures the similarity between $\mathcal{G}^{(i)}$ and $\mathcal{G}^*$ in terms of the Hamming distance. The Hamming distance in this context is the number of edge mismatches between two DAGs. That is, if we have two DAGs $\mathcal{G}, \mathcal{H}$ with the same number of nodes, we define

$$\|\mathcal{G} - \mathcal{H}\| = \sum_{i \neq j} \mathbb{1}\{\mathcal{G}_{ij} \neq \mathcal{H}_{ij}\}. \tag{2.31}$$

To be formally precise, we compute the Hamming distance between the adjacency matrices $\mathcal{A}_\mathcal{G}$ and $\mathcal{A}_\mathcal{H}$ of the DAGs $\mathcal{G}, \mathcal{H}$. However, to simplify the notation in this paper, when comparing two DAGs or networks in general, we use their graph names to implicitly refer also to their adjacency matrices.

The normalization term $Z(\beta^{(i)}, \mathcal{G}^*) = \sum_{\mathcal{G}^{(i)} \in \mathbb{M}} e^{-\beta^{(i)}||\mathcal{G}^{(i)} - \mathcal{G}^*||}$ in 2.30, where $\mathbb{M}$ denotes the space of all DAGs, is called the partition function and we further analyze it in Sections 3.3 and 3.4. The hyperparameter $\beta^{(i)}$ controls the strength of the coupling between $\mathcal{G}^{(i)}$ and $\mathcal{G}^*$. We assign higher probabilities to networks $\mathcal{G}^{(i)}$ that are similar to $\mathcal{G}^*$, larger values of $\beta^{(i)}$ further emphasize the similarity, while a smaller $\beta^{(i)}$ permits greater deviation, providing flexibility to fit the data when the hypernetwork may not fully capture the underlying structure.

Rooted in statistical mechanics, the Gibbs distribution describes the probability of a system being in a particular state based on its energy and a temperature parameter. In our context of probabilistic modelling, the "energy" corresponds to a measure of dissimilarity to the hypernetwork $\mathcal{G}^*$, and the distribution defines a probability over possible configurations, favoring those with lower energy [20]. The Gibbs distribution is a specific instance of the broader concept of a Gibbs measure, which originates from thermodynamics and describes how systems tend to settle into low-energy configurations [57]. Additionally, the Gibbs distribution plays a fundamental role in Gibbs sampling, a particular type of MCMC method used for posterior modelling in probabilistic inference [24].

The limiting behavior of the Gibbs prior 2.30 was briefly discussed in [67], but here we explore it more explicitly. When $\beta^{(i)} \to 0$, we have

$$\lim_{\beta^{(i)} \to 0} \frac{e^{-\beta^{(i)}\|\mathcal{G}^{(i)} - \mathcal{G}^*\|}}{Z(\beta^{(i)}, \mathcal{G}^*)} \propto \frac{1}{c}, \text{ where } c \in \mathbb{R}, c > 0, \tag{2.32}$$

leads to a diffuse, uninformative prior over the support[†] of $\mathcal{G}^{(i)}$, meaning that the coupling is completely "turned off". Consequently, the resulting conditional posterior probability of the model would be proportional to the likelihood terms $\mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)})$.

[†]Support of a function $f\colon \text{dom}(f) \subset X \to \mathbb{R}$ is defined as $\text{supp}(f) = \{x \in \text{dom}(f) \subset X \mid f(x) \neq 0\}$ [51].

On the other hand, if $\beta^{(i)} \to \infty$, all the terms $e^{-\beta^{(i)}\|\mathcal{G}-\mathcal{G}^*\|}$ in $Z(\beta^{(i)}, \mathcal{G}^*)$ goes to 0 if $\mathcal{G} \neq \mathcal{G}^*$, while $e^{-\beta^{(i)}\|\mathcal{G}-\mathcal{G}^*\|} = 1$ if $\mathcal{G} = \mathcal{G}^*$. Hence, we obtain

$$\lim_{\beta^{(i)}\to\infty} \frac{e^{-\beta^{(i)}\|\mathcal{G}^{(i)}-\mathcal{G}^*\|}}{Z(\beta^{(i)}, \mathcal{G}^*)} = \mathbb{1}\{\mathcal{G}^{(i)} = \mathcal{G}^*\}, \tag{2.33}$$

where we use the indicator function to denote the sharp density peak of the prior.

2.5.1 General MCMC Scheme

In the Bayesian context, if we want to infer a network $\mathcal{G}$ from data $\mathcal{D}$, we are aiming to find a network structure that maximizes the posterior probability $\mathbb{P}(\mathcal{G} \mid \mathcal{D})$. We have

$$\mathbb{P}(\mathcal{G} \mid \mathcal{D}) \propto \mathbb{P}(\mathcal{D} \mid \mathcal{G})\mathbb{P}(\mathcal{G}), \tag{2.34}$$

where $\mathbb{P}(\mathcal{D} \mid \mathcal{G})$ is the evidence and $\mathbb{P}(\mathcal{G})$ is the prior network structures distribution. However, as we shall see in the next sections, the number of possible network structures grows superexponenitally with the number of nodes and this complicates solving the optimization problem of maximizing 2.34. Moreover, it is uncommon in systems biology to have enough data (which is often very sparse) to infer a single network that sharply defines the posterior $\mathbb{P}(\mathcal{G} \mid \mathcal{D})$, especially given that the network is meant to capture complex interactions. Instead, the posterior distribution is typically diffused across many possible network configurations. Sampling algorithms can capture such a diffusely spread distribution by providing us with a sequence of high-scoring networks, and the average of the sampled networks can be used for inference. Although direct sampling from 2.34 is intractable, we can employ the Metropolis-Hastings algorithm [44, 29], which defines the MCMC method [35].

Application of MCMC for learning Bayesian networks was first introduced in [41]. We want to model the posterior $\mathbb{P}(\mathcal{G} \mid \mathcal{D})$ using samples $\mathcal{G}_1, \mathcal{G}_2, \ldots$ coming from a Markov chain with a transition kernel $T(\mathcal{G}_i \mid \mathcal{G}_k)$, and a probability distribution $\mathbb{P}_n(\mathcal{G}_k)$ in the nth step of the algorithm,

$$\mathbb{P}_{n+1}(\mathcal{G}_i) = \sum_k T(\mathcal{G}_i \mid \mathcal{G}_k)\mathbb{P}_n(\mathcal{G}_k). \tag{2.35}$$

Recall from Section 2.2 that a discrete and ergodic Markov chain converges, in distribution, to its stationary distribution $\mathbb{P}_\infty(\mathcal{G}_k)$,

$$\mathbb{P}_n(\mathcal{G}_k) \xrightarrow{n\to\infty} \mathbb{P}_\infty(\mathcal{G}_k).$$

The goal is to construct a transition kernel matrix $\mathbf{T}$ such that the stationary distribution of the corresponding Markov chain coincides with the posterior probability,

$$\mathbb{P}_\infty(\mathcal{G}) = \mathbb{P}(\mathcal{G} \mid \mathcal{D}). \tag{2.36}$$

It turns out that a sufficient condition for $\mathbf{T}$ is that it needs to satisfy the equation of detailed balance

$$\frac{T(\mathcal{G}_k \mid \mathcal{G}_i)}{T(\mathcal{G}_i \mid \mathcal{G}_k)} = \frac{\mathbb{P}(\mathcal{G}_k \mid \mathcal{D})}{\mathbb{P}(\mathcal{G}_i \mid \mathcal{D})} = \frac{\mathbb{P}(\mathcal{D} \mid \mathcal{G}_k)\mathbb{P}(\mathcal{G}_k)}{\mathbb{P}(\mathcal{D} \mid \mathcal{G}_i)\mathbb{P}(\mathcal{G}_i)}. \tag{2.37}$$

Assuming 2.37 holds, we have

$$\begin{aligned} T(\mathcal{G}_k \mid \mathcal{G}_i)\mathbb{P}(\mathcal{G}_i \mid \mathcal{D}) &= T(\mathcal{G}_i \mid \mathcal{G}_k)\mathbb{P}(\mathcal{G}_k \mid \mathcal{D}) \\ &\Longrightarrow \\ \sum_k T(\mathcal{G}_i \mid \mathcal{G}_k)\mathbb{P}(\mathcal{G}_i \mid \mathcal{D}) &= \sum_k T(\mathcal{G}_k \mid \mathcal{G}_i)\mathbb{P}(\mathcal{G}_i \mid \mathcal{D}) \\ &= \mathbb{P}(\mathcal{G}_i \mid \mathcal{D}) \sum_k T(\mathcal{G}_k \mid \mathcal{G}_i) \\ &= \mathbb{P}(\mathcal{G}_i \mid \mathcal{D}), \end{aligned} \tag{2.38}$$

where the last equality holds because $\mathcal{G}_1, \ldots, \mathcal{G}_k, \ldots$ are partitioning the sample space. In practice, the transition $\mathcal{G}_k \to \mathcal{G}_i$ is split into two steps. First, we propose a new state from a proposal distribution

$Q(\mathcal{G}_i \mid \mathcal{G}_k)$ and the new structure $\mathcal{G}_i$ is accepted with probability $A(\mathcal{G}_i \mid \mathcal{G}_k)$. Therefore, $T(\mathcal{G}_i \mid \mathcal{G}_k) = Q(\mathcal{G}_i \mid \mathcal{G}_k)A(\mathcal{G}_i \mid \mathcal{G}_k)$. Using the equation of detailed balance, we obtain

$$\begin{aligned} \frac{A(\mathcal{G}_k \mid \mathcal{G}_i)}{A(\mathcal{G}_i \mid \mathcal{G}_k)} &= \frac{\mathbb{P}(\mathcal{D} \mid \mathcal{G}_k)\mathbb{P}(\mathcal{G}_k)Q(\mathcal{G}_i \mid \mathcal{G}_k)}{\mathbb{P}(\mathcal{D} \mid \mathcal{G}_i)\mathbb{P}(\mathcal{G}_i)Q(\mathcal{G}_k \mid \mathcal{G}_i)} \\ &\Longrightarrow \\ A(\mathcal{G}_k \mid \mathcal{G}_i) &= \min\left\{\frac{\mathbb{P}(\mathcal{D} \mid \mathcal{G}_k)\mathbb{P}(\mathcal{G}_k)Q(\mathcal{G}_i \mid \mathcal{G}_k)}{\mathbb{P}(\mathcal{D} \mid \mathcal{G}_i)\mathbb{P}(\mathcal{G}_i)Q(\mathcal{G}_k \mid \mathcal{G}_i)}, 1\right\}. \end{aligned} \tag{2.39}$$

2.5.2 Model Specific MCMC Scheme

Going back to our coupled probabilistic model 2.4, the idea is to sample the individual network structures $\mathcal{G}^{(i)}$, the hyperparameters $\beta^{(i)}$ and the hypernetwork $\mathcal{G}^*$ from the posterior distribution. Suppose we have a DAG $\mathcal{G}^{(i)}_{\text{old}}$. We propose a new DAG $\mathcal{G}^{(i)}_{\text{new}}$ from the proposal distribution $Q(\mathcal{G}^{(i)}_{\text{new}} \mid \mathcal{G}^{(i)}_{\text{old}})$. Similarly, let $W(\mathcal{G}^*_{\text{new}} \mid \mathcal{G}^*_{\text{old}})$ be the proposal distribution for the hypernetworks, and $R(\beta^{(i)}_{\text{new}} \mid \beta^{(i)}_{\text{old}})$ be a proposal distribution for the hyperparameters. For proposing the DAGs $\mathcal{G}^{(i)}$ and $\mathcal{G}^*$, we use the following elementary edge-based operations: addition, deletion, or reversal of an edge. We propose new networks with an equal chance over the neighboring networks,

$$Q(\mathcal{G}^{(i)}_{\text{new}} \mid \mathcal{G}^{(i)}_{\text{old}}) = \frac{\mathbb{1}\{\mathcal{G}^{(i)}_{\text{new}} \in \mathcal{N}(\mathcal{G}^{(i)}_{\text{old}})\}}{|\mathcal{N}(\mathcal{G}^{(i)}_{\text{old}})|}, \text{ and } W(\mathcal{G}^*_{\text{new}} \mid \mathcal{G}^*_{\text{old}}) = \frac{\mathbb{1}\{\mathcal{G}^*_{\text{new}} \in \mathcal{N}(\mathcal{G}^*_{\text{old}})\}}{|\mathcal{N}(\mathcal{G}^*_{\text{old}})|}. \tag{2.40}$$

We assume that $R(\beta^{(i)}_{\text{new}} \mid \beta^{(i)}_{\text{old}})$ is symmetric, so $R(\beta^{(i)}_{\text{new}} \mid \beta^{(i)}_{\text{old}}) = R(\beta^{(i)}_{\text{old}} \mid \beta^{(i)}_{\text{new}})$. More details about the choice of the prior for the hyperparameters can be found in Section 4.2. Using the conditional independence given by the joint probability 2.30, the symmetric property of the proposal distribution $R(\beta^{(i)}_{\text{new}} \mid \beta^{(i)}_{\text{old}})$, and 2.40, the acceptance probability in the Metropolis-Hastings rule takes the form

$$\begin{aligned} A &= \min\left\{\prod_{i=1}^{I} \frac{\mathbb{P}(\mathcal{D}_i, \mathcal{G}^{(i)}_{\text{new}}, \beta^{(i)}_{\text{new}}, \mathcal{G}^*_{\text{new}})Q_i(\mathcal{G}^{(i)}_{\text{old}} \mid \mathcal{G}^{(i)}_{\text{new}})R_i(\beta^{(i)}_{\text{old}} \mid \beta^{(i)}_{\text{new}})W(\mathcal{G}^*_{\text{old}} \mid \mathcal{G}^*_{\text{new}})}{\mathbb{P}(\mathcal{D}_i, \mathcal{G}^{(i)}_{\text{old}}, \beta^{(i)}_{\text{old}}, \mathcal{G}^*_{\text{old}})Q_i(\mathcal{G}^{(i)}_{\text{new}} \mid \mathcal{G}^{(i)}_{\text{old}})R_i(\beta^{(i)}_{\text{new}} \mid \beta^{(i)}_{\text{old}})W(\mathcal{G}^*_{\text{new}} \mid \mathcal{G}^*_{\text{old}})}, 1\right\} \\ &= \min\left\{\prod_{i=1}^{I} \frac{\mathbb{P}(\mathcal{D}_i, \mathcal{G}^{(i)}_{\text{new}}, \beta^{(i)}_{\text{new}}, \mathcal{G}^*_{\text{new}})Q_i(\mathcal{G}^{(i)}_{\text{old}} \mid \mathcal{G}^{(i)}_{\text{new}})W(\mathcal{G}^*_{\text{old}} \mid \mathcal{G}^*_{\text{new}})}{\mathbb{P}(\mathcal{D}_i, \mathcal{G}^{(i)}_{\text{old}}, \beta^{(i)}_{\text{old}}, \mathcal{G}^*_{\text{old}})Q_i(\mathcal{G}^{(i)}_{\text{new}} \mid \mathcal{G}^{(i)}_{\text{old}})W(\mathcal{G}^*_{\text{new}} \mid \mathcal{G}^*_{\text{old}})}, 1\right\} \\ &= \min\left\{\prod_{i=1}^{I} \frac{\mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)}_{\text{new}})\mathbb{P}(\mathcal{G}^{(i)}_{\text{new}} \mid \beta^{(i)}_{\text{new}}, \mathcal{G}^*_{\text{new}})|\mathcal{N}(\mathcal{G}^{(i)}_{\text{old}})||\mathcal{N}(\mathcal{G}^*_{\text{old}})|}{\mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)}_{\text{old}})\mathbb{P}(\mathcal{G}^{(i)}_{\text{old}} \mid \beta^{(i)}_{\text{old}}, \mathcal{G}^*_{\text{old}})|\mathcal{N}(\mathcal{G}^{(i)}_{\text{new}})||\mathcal{N}(\mathcal{G}^*_{\text{new}})|}, 1\right\}. \end{aligned} \tag{2.41}$$

The terms $\mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)})$ are computed using the BGe score. To improve the acceptance probability and thereby enhance the mixing and convergence of the Markov chain, it is proposed in [67] to divide A into smaller submoves $A(\mathcal{G}^{(i)}_{\text{new}} \mid \mathcal{G}^{(i)}_{\text{old}})$, $A(\beta^{(i)}_{\text{new}} \mid \beta^{(i)}_{\text{old}})$, and $A(\mathcal{G}^*_{\text{new}} \mid \mathcal{G}^*_{\text{old}})$. These submoves are repeated in a loop as follows. First, for $i = 1, \ldots, I$, we compute

$$\begin{aligned} A(\mathcal{G}^{(i)}_{\text{new}} \mid \mathcal{G}^{(i)}_{\text{old}}) &= \min\left\{\frac{\mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)}_{\text{new}})\mathbb{P}(\mathcal{G}^{(i)}_{\text{new}} \mid \beta^{(i)}, \mathcal{G}^*)|\mathcal{N}(\mathcal{G}^{(i)}_{\text{old}})|}{\mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)}_{\text{old}})\mathbb{P}(\mathcal{G}^{(i)}_{\text{old}} \mid \beta^{(i)}, \mathcal{G}^*)|\mathcal{N}(\mathcal{G}^{(i)}_{\text{new}})|}, 1\right\} \\ &= \min\left\{\frac{\mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)}_{\text{new}})e^{-\beta^{(i)}\|\mathcal{G}^{(i)}_{\text{new}} - \mathcal{G}^*\|}|\mathcal{N}(\mathcal{G}^{(i)}_{\text{old}})|}{\mathbb{P}(\mathcal{D}_i \mid \mathcal{G}^{(i)}_{\text{old}})e^{-\beta^{(i)}\|\mathcal{G}^{(i)}_{\text{old}} - \mathcal{G}^*\|}|\mathcal{N}(\mathcal{G}^{(i)}_{\text{new}})|}, 1\right\}, \end{aligned} \tag{2.42}$$

while keeping $\beta^{(i)}$'s and $\mathcal{G}^*$ fixed. Next, we similarly set

$$A(\beta^{(i)}_{\text{new}} \mid \beta^{(i)}_{\text{old}}) = \min\left\{\frac{\mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}_{\text{new}}, \mathcal{G}^*)}{\mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}_{\text{old}}, \mathcal{G}^*)}, 1\right\}, \tag{2.43}$$

where we use the current values of $\mathcal{G}^{(i)}$ (that could be accepted or rejected based on $A(\mathcal{G}^{(i)}_{\text{new}} \mid \mathcal{G}^{(i)}_{\text{old}})$) and a fixed hypernetwork $\mathcal{G}^*$. Using the new/old networks and their hyperparameters, we finally compute the acceptance ratio for the hypernetwork as

$$A(\mathcal{G}^*_{\text{new}} \mid \mathcal{G}^*_{\text{old}}) = \min\left\{\prod_{i=1}^{I} \frac{\mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}, \mathcal{G}^*_{\text{new}})|\mathcal{N}(\mathcal{G}^*_{\text{old}})|}{\mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}, \mathcal{G}^*_{\text{old}})|\mathcal{N}(\mathcal{G}^*_{\text{new}})|}, 1\right\}. \tag{2.44}$$

To accept a new state with probability A we can draw a uniform number $u \sim \mathcal{U}[0, 1]$, and accept the new state if $A > u$ or reject if $A \leq u$.

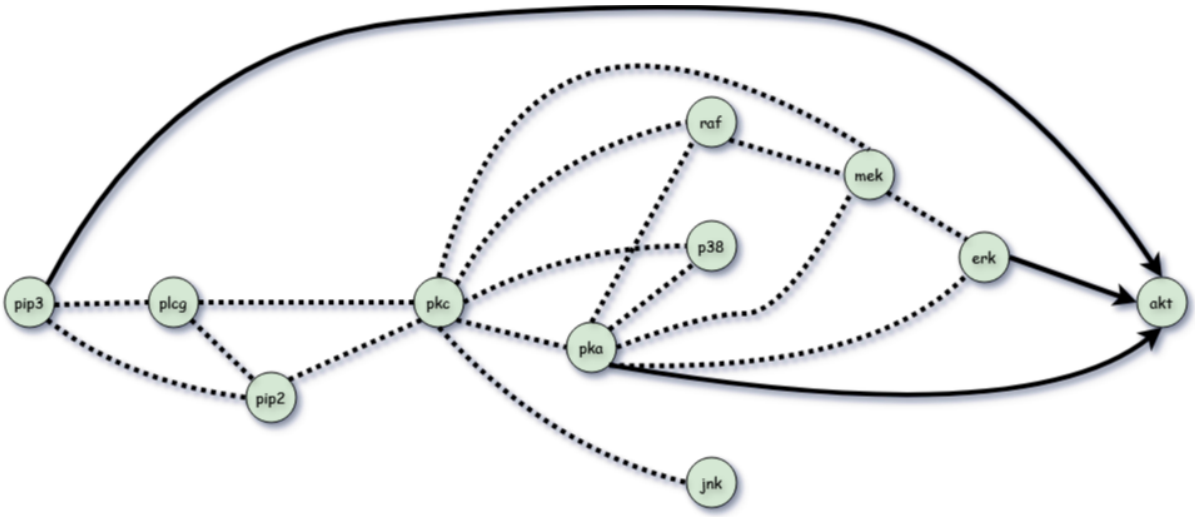


Figure 3.1: CPDAG of the gold-standard Raf pathway. Undirected edges are shown as dotted lines. Notice that there are only 3 compelled edges. The figure was created using Draw.io [22].

3 Enhanced Model with New Coupling Schemes

This section summarizes our theoretical contributions. First, we extend the model from Section 2.5 with CPDAG coupling using Structural Hamming Distance (SHD) [62, 2], and we present an explicit equation that is suitable for implementation in any scripting language. We then introduce a second coupling mechanism based on skeletons, where mismatches are measured by the classical Hamming distance. In Section 3.3, we propose an enhanced version of a standard approximation scheme from statistical physics. To do this, we quantify the possible sizes and multiplicities of mismatches between the true parent configuration of a given node in the DAG and a set of proposed parent configurations under a fan-in restriction. For that purpose, we formally define a concatenation operation, which enables pre-computation and storage of a complete characterization of all possible mismatches in a DAG, thereby improving the computational efficiency of the MCMC scheme described in Section 2.5.2. Algorithmic Schemes 3.1, 3.2, and 3.3 summarize this approach. Finally, in Section 3.4, we discuss the corrections required for the partition function when integrating our two new coupling schemes into the model of Section 2.5. In particular, we propose a method to quantify and measure mismatches between two parent-node configurations that include both directed and undirected edges.

3.1 CPDAG Coupling

One aspect of the model described in Section 2.5 can be perceived as problematic. The Hamming distance $\|\mathcal{G} - \mathcal{G}^*\|$ measures the similarity between $\mathcal{G}$ and $\mathcal{G}^*$ in their DAG space. This penalizes structures that are equivalent in their CPDAG space and, therefore, have the same BGe score. In general, and without making strong assumptions, it is impossible to identify or learn the true causal DAG, even with an infinite amount of data [49]. For example, when trying to reconstruct the gold-standard Raf pathway, inferring its true structure is impossible because the CPDAG of Raf contains only 3 compelled edges, as can be seen in Figure 3.1. It is, therefore, realistic to learn only the equivalence classes of Raf and/or its underlying skeleton structure[†]. Hence, we want to take into account the equivalence classes during network learning.

The first attempts to develop metrics for comparing DAGs based on the patterns of their equivalence classes were introduced in [60] in the context of the PC algorithm. In the PC algorithm, mismatches between DAGs are assessed at the equivalence class level, taking into account undirected edges and inherent ambiguities in causal directionality. Further improvements of the PC algorithm were proposed in [1] and in [8]. More recent approaches explicitly introducing distance metrics between CPDAGs include [48] with Structural Intervention Distance, [53] with Causal Edit Distance, or [31] with various

[†]Also, as a given CPDAG encodes equivalence classes of various DAGs, intuitively, one expects that sampling CPDAGs instead of DAGs in the MCMC scheme can capture more information about the network structure of interest, although in general, the improvement appears rather conservative [15].

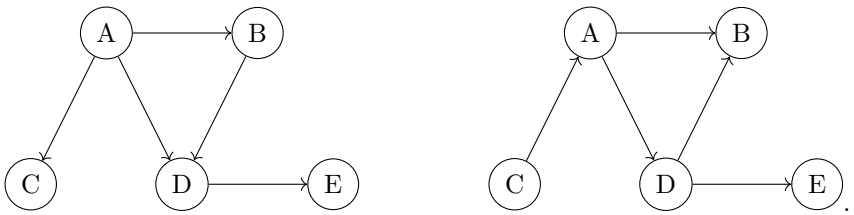
Identification Distances.

Perhaps the most intuitive metric in comparing the similarity between two CPDAGs is the SHD. The SHD between two CPDAGs $\mathcal{G}$ and $\mathcal{G}^*$, denoted as $\|\mathcal{G} - \mathcal{G}^*\|_{\text{shd}}$ is the minimal number of single edge operations that are needed to transfer $\mathcal{G}$ into $\mathcal{G}^*$. The single edge operations are: add or delete an undirected edge; add, delete, or reverse a directed edge; add or delete the orientation of an edge.

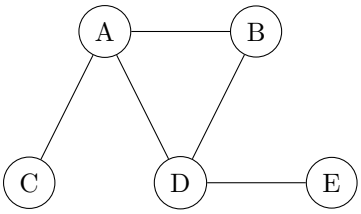
With a little abuse of the notation, assume that we have a CPDAG with an adjacency matrix $\mathcal{G}$ and another CPDAG with an adjacency matrix $\mathcal{G}^*$, both with the same number of nodes N. If there is a directed edge from a node i to j, we write $\mathcal{G}_{ij} = 1$. If the edge connecting nodes i and j does not have a direction, we write $\mathcal{G}_{ij} = -1$, and if there is no edge between i and j, we write $\mathcal{G}_{ij} = 0$. Similarly for $\mathcal{G}^*$. Thus, $\mathcal{G}, \mathcal{G}^* \in \{-1, 0, 1\}^{N \times N}$, where N is the number of nodes. We present a new compact way to calculate the SHD that requires traversing only through half of the entries of the matrices $\mathcal{G}$ and $\mathcal{G}^*$. When computing the SHD $\|\mathcal{G} - \mathcal{G}^*\|_{\text{shd}}$, we count the mismatches between pairs of nodes (i, j) in both graphs. To ensure that we avoid duplicates, we do this for $i < j$. Hence, we do the following,

$$\|\mathcal{G} - \mathcal{G}^*\|_{\text{shd}} = \sum_{1 \leq i < j \leq N} \mathbb{1}\{(\mathcal{G}_{ij}, \mathcal{G}_{ji}) \neq (\mathcal{G}^*_{ij}, \mathcal{G}^*_{ji})\}. \tag{3.1}$$

To illustrate the SHD, consider the following two DAGs,

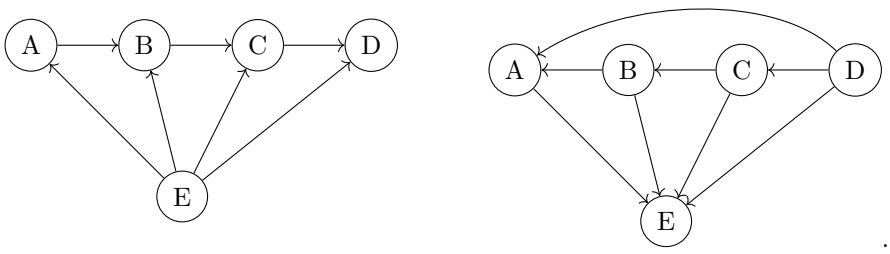


The Hamming distance between the graphs is 2. However, both graphs share the same CPDAG,

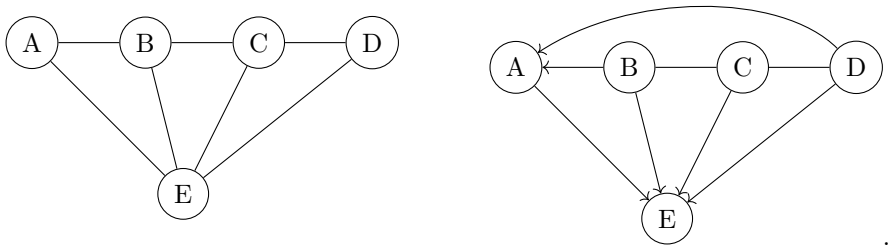


Thus, both graphs belong to the same equivalence class. This makes them statistically indistinguishable from each other so it is questionable whether we should penalize mismatches that practically do not matter for computing their scores. On the other hand, the Structural Hamming distance between the two graphs is 0.

For a second example, consider the DAGs,



The Hamming distance between the graphs is equal to 8. By transferring each DAG to its CPDAG, we obtain



The Structural Hamming Distance between those two CPDAGs is equal to 6.

Going back to the Gibbs prior 2.30 for the network structures in the coupled model proposed by Werhli & Husmeier, we replace the Hamming distance in the equation with the Structural Hamming distance:

$$\mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}, \mathcal{G}^*) = \frac{e^{-\beta^{(i)}\|\mathcal{G}^{(i)}-\mathcal{G}^*\|_{\text{shd}}}}{Z(\beta^{(i)}, \mathcal{G}^*)}, \text{ for } i = 1, \ldots, I. \tag{3.2}$$

Remark 1 We can still use the same MCMC method described in Section 2.5.2 to sample DAGs in implementing this proposed CPDAG coupling. It is sufficient to convert the network structures $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(I)}, \mathcal{G}^*$ to CPDAGs before entering the Gibbs distribution.

Remark 2 Notice that with the CPDAG coupling, also the partition function $Z(\beta^{(i)}, \mathcal{G}^*)$ changes because the input structure is now a CPDAG. How we deal with this issue is explained in Section 3.4.

3.2 Skeleton Coupling

Instead of looking at the direction or orientation of edges, we may compare two DAGs $\mathcal{G}$ and $\mathcal{G}^*$ purely based on their underlying skeleton. While using this approach in the context of a gold-standard Raf pathway, we lower the chance of correctly learning the 3 compelled edges, we do not penalize any wrong orientations of edges that, in principle, form the true skeleton. Thus, instead of computing the Hamming distance between DAGs, we compute the Hamming distance between their skeletons.

Notice that given an adjacency matrix $\mathcal{G}$ of a DAG, we can recover its skeleton (equivalently seen as adding the opposite direction to every edge in $\mathcal{G}$) by adding its transpose to it: $\mathcal{G} + \mathcal{G}^T$. Thus, we calculate the Hamming distance between the skeletons of two DAGs $\mathcal{G}$ and $\mathcal{G}^*$ on N nodes as follows

$$\|\mathcal{G} - \mathcal{G}^*\|_{\text{skel}} = \sum_{1 \le i < j \le N} \mathbb{1}\{\mathcal{G}_{ij} + \mathcal{G}_{ij}^T \neq \mathcal{G}_{ij}^* + (\mathcal{G}_{ij}^*)^T\}, \tag{3.3}$$

where we sum over all $i < j$ to avoid duplicates.

The Gibbs prior 2.30 thus becomes

$$\mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}, \mathcal{G}^*) = \frac{e^{-\beta^{(i)}\|\mathcal{G}^{(i)}-\mathcal{G}^*\|_{\text{skel}}}}{Z(\beta^{(i)}, \mathcal{G}^*)}, \text{ for } i = 1, \ldots, I. \tag{3.4}$$

As noted in Remark 2, the partition function has to change accordingly. We propose a way to deal with this issue in Section 3.4.

3.3 Efficient Computation of the Partition Function

Assume that $\mathcal{G}^*$ and $\mathcal{G}$'s are DAGs on N nodes and consider the Hamming distance $||\cdot||$. The partition function

$$Z = Z(\beta, \mathcal{G}^*) = \sum_{\mathcal{G} \in \mathbb{M}} e^{-\beta||\mathcal{G}-\mathcal{G}^*||}, \tag{3.5}$$

which acts as a normalizing function in the Gibbs prior 2.30, sums over all DAGs $\mathbb{M}$ on N nodes. We are focusing primarily on the hypernetwork $\mathcal{G}^*$, so we write β instead of $\beta^{(i)}$ throughout this section as emphasizing the individual (sub)networks $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(I)}$ is not particularly important here.

The exact enumeration of DAGs was first characterized in [54, 61]. If A_N is the number of DAGs on N nodes, the following recursive formula holds for $N \geq 1$[†]

$$A_N = \sum_{k=1}^{N} (-1)^{k+1} \binom{N}{k} 2^{k(N-k)} A_{N-K}, \text{ with } A_0 = 1. \tag{3.6}$$

The asymptotic behavior is

$$A_N \approx N! \frac{2^{\binom{N}{2}}}{Mp^N}, \text{ where } p = 1.488\ldots \text{ and } M = 0.474\ldots, \tag{3.7}$$

which means that the number of DAGs grows superexponentially. Consequently, computing the partition function using Eq. 3.5 is unrealistic, especially for larger networks.

We present two steps for reducing the computational complexity when computing the partition function Z - one that is well-known and one that we propose. It is suggested in [66, 67] to use the Perfect Gas Approximation (PGA), popular in statistical physics:

$$Z = Z(\beta, \mathcal{G}^*) \approx \prod_{n=1}^{N} \sum_{\pi_n} e^{-\beta \mathcal{E}(n, \pi_n)}, \tag{3.8}$$

where we sum over all possible parent configurations π_n of a node n and

$$\mathcal{E}(n, \pi_n) = \sum_{i \in \pi_n} (1 - \mathcal{G}^*_{in}) + \sum_{i \notin \pi_n} \mathcal{G}^*_{in}. \tag{3.9}$$

The motivation is as follows. For a fixed hypernetwork $\mathcal{G}^*$, we can see the Hamming distance $||\mathcal{G} - \mathcal{G}^*||$ in Eq. 3.5 as the "energy" $E(\mathcal{G})$ of a network $\mathcal{G}$. The term $\mathcal{E}(n, \pi_n)$ counts the mismatches between π_n and the true set of parents of a node n in $\mathcal{G}^*$, $\pi_n[\mathcal{G}^*]$. In other words, it is the (local) Hamming distance between the substructures induced by elements in π_n and $\pi_n[\mathcal{G}^*]$. Thus, we can write

$$||\mathcal{G} - \mathcal{G}^*|| = E(\mathcal{G}) = \sum_{n=1}^{N} \mathcal{E}(n, \pi_n[\mathcal{G}]). \tag{3.10}$$

Hence,

$$Z = \sum_{\mathcal{G} \in \mathbb{M}} e^{-\beta E(\mathcal{G})} = \sum_{\pi_1} \times \cdots \times \sum_{\pi_N} e^{-\beta(\mathcal{E}(1,\pi_1) + \cdots + \mathcal{E}(N,\pi_N))} \tag{3.11}$$

$$= \prod_{n=1}^{N} \sum_{\pi_n} e^{-\beta \mathcal{E}(n, \pi_n)}. \tag{3.12}$$

The PGA is, in practical computations, an approximation method. We constrain the cardinality of parent configurations by imposing a fan-in restriction F on the number of parents each node in the (hyper)network can have. This means that we only consider networks where each node has at most F parents. While this approach significantly reduces computational complexity, it comes at a cost. Modifications of a parent configuration π_n can sometimes introduce directed cycles, rendering the network structure invalid. This, in turn, leads to a slight overestimation of the partition function Z [33]. However, according to [66], the carried-over bias is not critical. A reasonable choice is to set $F = 3$, which ensures that densely connected networks are not sampled by the MCMC scheme. Consequently, the number of invalid cyclic structures remains small.

Next, we propose a way to even more efficiently compute the approximation of $Z = Z(\beta, \mathcal{G}^*)$. Given a hypernetwork $\mathcal{G}^*$ defined on N nodes, the fan-in restriction F, the range of $\mathcal{E}$ defined in Eq. 3.9 has finite size. An arbitrary node n of $\mathcal{G}^*$ has exactly

$$k_n = \sum_{i=1}^{N} \mathbb{1}\{\mathcal{G}^*_{in} = 1\} \tag{3.13}$$

[†]Interestingly, there is a bijection between A_N and $N \times N$ binary matrices with positive real eigenvalues [43].

Algorithm 3.1 Sequence Accumulating k_n-configuration Interactions (SAKI)

N: Number of nodes in a DAG $\mathcal{G}^*$; F: Fan-in restriction; k_n: Number of parents of a node n in the DAG $\mathcal{G}^*$.
procedure SAKI(N, F, k_n)
    $\xi^{(k_n)} \leftarrow ()$;
    for $f \in \{0, \dots, F\}$ **do**
        for $j \in \{0, \dots, \min(k_n, f)\}$ **do**
            $S \leftarrow \binom{k_n}{j}\binom{N-1-k_n}{f-j}$;
            $L(f, j) \leftarrow ()$;
            for $l \in \{1, \dots, S\}$ **do**
                $L(f, j) \leftarrow L(f, j) \oplus ((k_n - j) + (f - j))$;
            end for
            $\xi^{(k_n)} \leftarrow \xi^{(k_n)} \oplus L(f, j)$;
        end for
    end for
    Sort the sequence ξ^{k_n} in ascending order;
    Return ξ^{k_n}
end procedure

parents. Assume that $A = \{a_1, \dots, a_{k_n}\}$ is the true parent configuration of a node n and pick an arbitrary $B = \{b_1, \dots, b_f\} \subset \pi_n, 0 \leq f \leq F$[†]. We define the mismatch between A and B as the elements belonging to either A or B, but not to their intersection. The symmetric difference captures this, $A \Delta B = (A \setminus B) \cup (B \setminus A)$. In general, we have

$$|A \Delta B| = (k_n - j) + (f - j), \text{ for some } f \in \{0, \dots, F\} \text{ and } j \in \{0, \dots, \min(k_n, f)\}. \tag{3.14}$$

In addition to counting possible mismatch sizes, counting their multiplicities is also a deterministic combinatorial problem. Picking some $f \in \{0, \dots, F\}$ and $j \in \{0, \dots, \min(k_n, f)\}$, the mismatch of size $(k_n - j) + (f - j)$ occurs exactly $\binom{k_n}{j}\binom{N-1-k_n}{f-j}$-times when iterating through every $B \subset \pi_n$ such that $|B| = f$. To efficiently exploit this observation, we propose the following definition.

Definition 3.1 *Let $S_1, S_2, \dots, S_n$ be finite sequences $S_i = (s_{i,1}, s_{i,2}, \dots, s_{i,m_i})$ of length m_i, for $i = 1, \dots, n$. The **concatenation** of these sequences, denoted as $\bigoplus_{i=1}^{n} S_i = S_1 \oplus \cdots \oplus S_n$ is the sequence obtained by appending each sequence S_i to the end of the previous one,*

$$\bigoplus_{i=1}^{n} S_i = S_1 \oplus \cdots \oplus S_n = (s_{1,1}, s_{1,2}, \dots, s_{1,m_1}, s_{2,1}, s_{2,2}, \dots, s_{2,m_2}, \dots, s_{n,1}, s_{n,2}, \dots, s_{n,m_n}). \tag{3.15}$$

Using the concatenation from the definition above, let $S = S(N, k_n, f, j) = \binom{k_n}{j}\binom{N-1-k_n}{f-j}$ and denote

$$L(f, j) = (\underbrace{(k_n - j) + (f - j), (k_n - j) + (f - j), \dots}_{S = \binom{k_n}{j}\binom{N-1-k_n}{f-j}\text{-times}}) = \bigoplus_{l=1}^{S}((k_n - j) + (f - j)). \tag{3.16}$$

To obtain all the mismatches that can occur at a node n with k_n parents against proposed parent configurations of n, while respecting the fan-in restriction F, we concatenate the sequence $L(f, j)$ for every $f \in \{0, \dots, F\}$ and $j \in \{0, \dots, \min(k_n, f)\}$. Hence, set

$$\xi^{(k_n)} = \bigoplus_{f=0}^{F} \bigoplus_{j=0}^{\min(k_n, f)} L(f, j). \tag{3.17}$$

In other words, the sequence $\xi^{(k_n)}$ contains all the possible mismatch sizes, including their multiplicities, of an arbitrary node n in $\mathcal{G}^*$ which has exactly k_n parents. In addition, we require the entries of $\xi^{(k_n)}$ to be in ascending order. This is because every $\xi^{(k_n)}$ contains exactly one 0, which corresponds to the unique parent configuration from π_n that matches $\text{pa}_n[\mathcal{G}^*]$ and placing the 0 at the beginning makes

[†]The case $f = 0$ results in $B = \emptyset \in \pi_n$, which is also a valid parent configuration.

the approach of the next section feasible. We denote our approach for obtaining the sequence $\xi^{(k_n)}$ as Sequence Accumulating k_n-configuration Interactions (SAKI), which is summarized in the algorithmic scheme 3.1.

For example, if $F = 3$ and $N = 11$, we have

$$\begin{aligned}
\xi^{(0)} &= (0, \underbrace{1, 1, \ldots}_{\text{10-times}}, \underbrace{2, 2, \ldots}_{\text{45-times}}, \underbrace{3, 3, \ldots}_{\text{120-times}}), \\
\xi^{(1)} &= (0, \underbrace{1, 1, \ldots}_{\text{10-times}}, \underbrace{2, 2, \ldots}_{\text{45-times}}, \underbrace{3, 3, \ldots}_{\text{36-times}}, \underbrace{4, 4, \ldots}_{\text{84-times}}), \\
\xi^{(2)} &= (0, \underbrace{1, 1, \ldots}_{\text{10-times}}, \underbrace{2, 2, \ldots}_{\text{17-times}}, \underbrace{3, 3, \ldots}_{\text{64-times}}, \underbrace{4, 4, \ldots}_{\text{28-times}}, \underbrace{5, 5, \ldots}_{\text{56-times}}), \\
\xi^{(3)} &= (0, 1, 1, 1, \underbrace{2, 2, \ldots}_{\text{24-times}}, \underbrace{3, 3, \ldots}_{\text{22-times}}, \underbrace{4, 4, \ldots}_{\text{70-times}}, \underbrace{5, 5, \ldots}_{\text{21-times}}, \underbrace{6, 6, \ldots}_{\text{35-times}}).
\end{aligned}$$

Notice that $R = |\xi^{(k_n)}| = |\pi_n| = \sum_{f=0}^{F} \binom{N-1}{f}^{\dagger}$ for any n and k_n, respecting the fan-in restriction. It is however more convenient to treat $\xi^{(k_n)}$'s as vectors. Define the $(F+1) \times R$ matrix Ξ as

$$\Xi = \begin{bmatrix} \xi^{(0)} \\ \xi^{(1)} \\ \vdots \\ \xi^{(F)} \end{bmatrix}. \tag{3.18}$$

The matrix Ξ can be pre-computed and allocated in heap memory. We no longer have to compute the terms $\mathcal{E}(n, \pi_n)$ at each node, leading to further reduction of computational complexity of the PGA. Hence, the computation of the partition function becomes

$$Z = Z(\beta, \mathcal{G}^*) \approx \prod_{n=1}^{N} \sum_{x \in \xi^{(k_n)}} e^{-\beta x}, \tag{3.19}$$

where $k_n = \sum_{i=1}^{N} \mathbb{1}\{\mathcal{G}^*_{in} = 1\}$. We call the sum $\sum_{x \in \xi^{(k_n)}} e^{-\beta x}$ from 3.19 the Mismatch Energy Sum (MES). To emphasize its dependence on the parents of the node n, and the hyperparameter β, we also write it as $\text{MES}^{(n,\beta)}$.

We can also take into account the edge-based proposal operations. If we add a directed edge pointing towards n, we set $\xi^{(k_n)} := \xi^{(k_n+1)}$. If we delete a directed edge pointing towards n, we set $\xi^{(k_n)} := \xi^{(k_n-1)}$. If we invert a directed edge either pointing from n or pointing towards n, we set $\xi^{(k_n)} := \xi^{(k_n+1)}$ or $\xi^{(k_n)} := \xi^{(k_n-1)}$.

As the partition function also depends on the hyperparameter β, which is itself sampled within the MCMC scheme and thus varies throughout the iterations, any further reduction in the computational complexity of Eq. 3.19 is not feasible. The algorithmic scheme 3.3 provides a summary of the Enhanced Perfect Gas Approximation (EPGA), while the calculation of the matrix Ξ from 3.18 is detailed in 3.2. The general pipeline is: Algorithm 3.1 $\rightarrow$ Algorithm 3.2 $\rightarrow$ Algorithm 3.3.

3.4 Mismatch Energy Sums for each Coupling Scheme

The method of computing the partition function, as described in the previous section, is only valid when the hypernetwork $\mathcal{G}^*$ forms a proper DAG. For CPDAG coupling, the partition function is expected to be a CPDAG to ensure the proper normalization of the Gibbs prior 3.4. The same holds for the skeleton coupling. However, the computation of the partition function $Z = Z(\beta, \mathcal{G}^*)$ in these cases is far from straightforward. One major challenge arises from counting mismatches between the subgraph parent configurations in the given DAG and the proposed parent configurations. This task is relatively simple in the DAG case, as each edge has a fixed direction. Counting the mismatches imposes a simple combinatorial problem. However, for CPDAGs or skeletons, edges may be undirected or partially directed,

$^{\dagger}|\xi^{(k_n)}|$ denotes here the cardinality of the finite sequence $\xi^{(k_n)}$.

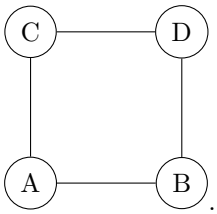
Algorithm 3.2 Precomputing Ξ

N: Number of nodes in a DAG $\mathcal{G}^*$; F: Fan-in restriction; X: Matrix $\begin{pmatrix} X_0 & \cdots & X_F \end{pmatrix}^T$ of size $(F+1) \times R = (F+1) \times \sum_{f=0}^{F} \binom{N-1}{f}$.
procedure XI_PRECOMPUTED(N, F, X)
  for $f \in \{0, \ldots, F\}$ **do**
    $\xi^{(f)} \leftarrow \text{SAKI}(N, F, f)$;
    Vectorize $\xi^{(f)}$;
    $X_f \leftarrow \xi^{(f)}$;
  end for
  Return $X = \begin{pmatrix} X_0 & \cdots & X_F \end{pmatrix}^T$
end procedure

Algorithm 3.3 Enhanced Perfect Gas Approximation (EPGA)

$\mathcal{G}^*$: DAG - hypernetwork; N: Number of nodes in $\mathcal{G}^*$; β: Coupling hyperparameter; $\Xi = \begin{pmatrix} \Xi_0 & \cdots & \Xi_F \end{pmatrix}^T = XI\_precomputed(N, F, X)$.
procedure EPGA($\mathcal{G}^*, N, \beta, \Xi$)
  $Z \leftarrow 1$;
  for $n \in \{1, \ldots, N\}$ **do**
    $k_n \leftarrow \sum_{i=1}^{N} \mathbb{1}\{\mathcal{G}^*_{in} = 1\}$;
    $\xi^{(k_n)} \leftarrow \Xi_{k_n}$;
    $\text{MES}^{(n,\beta)} \leftarrow 0$;
    for $x \in \{\xi_1^{(k_n)}, \ldots, \xi_R^{(k_n)}\}$ **do**
      $\text{MES}^{(n,\beta)} \leftarrow \text{MES}^{(n,\beta)} + e^{-\beta x}$;
    end for
    $Z \leftarrow Z \times \text{MES}^{(n,\beta)}$;
  end for
  Return Z
end procedure

allowing for multiple possible ways to model them. Another issue concerns the number of permissible network structures. For instance, consider the following undirected graph:



It is possible to construct 14 different DAGs while preserving the skeleton, each of which will fall into one of the 6 different equivalence classes - CPDAGs. See Figure 3.2. Thus, if we want to propose a way to also work with CPDAGs or skeletons in the partition function, our approach has to somehow take into account the fact that a given skeleton can be used to create various CPDAGs and DAGs. The acyclic orientations of a given undirected graph can be counted using the Tutte polynomial [6, 61]. However, computing the Tutte polynomial at every point is numerically challenging, and currently, approximation schemes exist only for sparse graphs [5].

When CPDAG and skeleton coupling are implemented, any undirected reversible edge can be equivalently seen as a bidirected edge. Therefore, X_1—X_2 can be seen as $X_1 \longleftrightarrow X_2$. We can denote the reversible edges with the -1 assigned to the corresponding adjacency matrix in a given network structure. Such an assignment is symmetric with respect to the diagonal of the adjacency matrix. By definition, all the edges in a fully undirected graph are reversible.[†]

Whether working with undirected or bidirected edges, the computation of $\mathcal{E}(n, \pi_n)$ becomes

[†]We emphasize that assigning the direction to the reversible edges is limited by the acyclicity constraint.

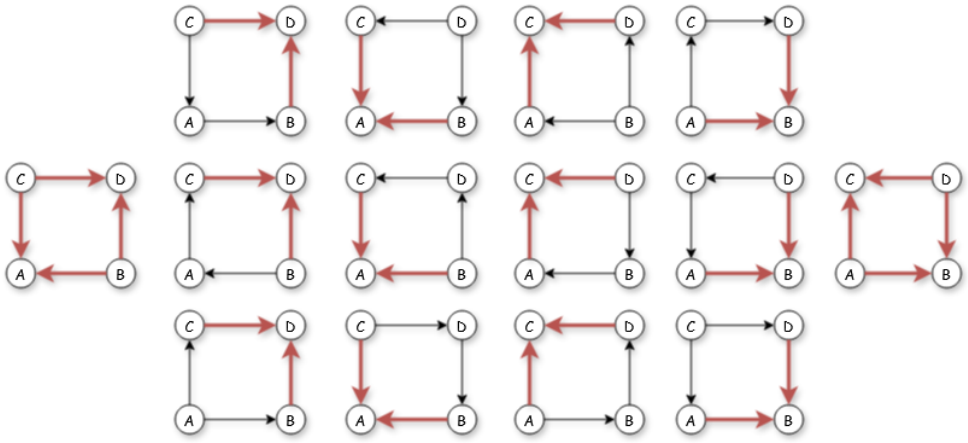


Figure 3.2: Acyclic orientations of a square-like skeleton graph. V-structures that determine the equivalence classes are highlighted in red. The figure was created using Draw.io [22].

complicated. If some parent configuration from π_n is proposed, the true parent configuration of the node n in CPDAG($\mathcal{G}^*$) or SKEL($\mathcal{G}^*$) may contain both compelled and reversible edges, which means that counting mismatches can be ambiguous. We propose a model that hopefully mitigates this ambiguity.

Recall the $\text{MES}^{(n,\beta)} = \sum_{x \in \xi^{(k_n)}} e^{-\beta x}$ from the previous section. When the hypernetwork $\mathcal{G}^*$ is a DAG, we have

$$\text{MES}^{(n,\beta)}_{\text{DAG}} = \sum_{x \in \xi^{(k_n)}} e^{-\beta x}, \text{ where } k_n = \sum_{i=1}^{N} \mathbb{1}\{\mathcal{G}^*_{in} = 1\} \text{ and } \xi^{(k_n)} \text{ is defined as in 3.17.} \tag{3.20}$$

Next, consider the fan-in restriction F. Easing the notation, when $\mathcal{G}^*$ is a CPDAG, define

$$k_n = \sum_{i=1}^{N} \mathbb{1}\{\mathcal{G}^*_{in} = 1\}, \tag{3.21}$$

$$l_n = \sum_{i=1}^{N} \mathbb{1}\{\mathcal{G}^*_{in} = -1\}, \text{ and} \tag{3.22}$$

$$\hat{\xi}^{(k_n,l_n)} = \frac{\sum_{j=k_n}^{\min(F,k_n+l_n)} \xi^{(j)}}{\min(F+1-k_n, l_n+1)}, \tag{3.23}$$

and define

$$\text{MES}^{(n,\beta)}_{\text{CPDAG}} = \begin{cases} \sum_{x \in \xi^{(k_n)}} e^{-\beta x}, & \text{if } l_n = 0, \\ \sum_{x \in \hat{\xi}^{(k_n,l_n)}} e^{-\beta x}, & \text{if } l_n \neq 0. \end{cases} \tag{3.24}$$

The motivation is as follows. Given a node n, we first count its parents k_n and the number l_n of the undirected edges attached to it. Thus, there are at least k_n edges pointing towards n, while each of the l_n undirected edges can be pointing towards n or from n to another node. The "true" mismatch vector can be any of $\xi^{(k_n)}, \xi^{(k_n+1)}, \ldots, \xi^{\min(F,k_n+l_n)}$, respecting the fan-in restriction F. Due to this uncertainty, it is a reasonable assumption that the mean $\hat{\xi}^{(k_n,l_n)}$ of the mismatch vectors $\xi^{(k_n)}, \xi^{(k_n+1)}, \ldots, \xi^{\min(F,k_n+l_n)}$ serves as an appropriate approximation of the "true" mismatch vector. Occasionally, deviations may occur in either direction; however, the error should remain small over the long term.

One may think of alternative approaches for defining the $\text{MES}^{(n)}_{\text{CPDAG}}$, for instance, by halving each mismatch vector $\xi^{(i)}$, for $i = 1, \ldots, \max(F, k_n + l_n)$ or even $i = 0, \ldots, \max(F, k_n + l_n)$ and concatenate $\tilde{\xi} = \xi^{(1)} \oplus \cdots \oplus \xi^{(\max(F,k_n+l_n))}$. Such an approach, however, does not work in real simulations. The exponential terms in the sum $\sum_{x \in \tilde{\xi}} e^{-\beta x}$ are evidently not sufficiently large, which pushes the sum to large values. The model tries to compensate for this by quickly sampling the hyperparameters β to maximum values, which consequently overvalue the effect of the coupling. The flexibility in the model

is thus lost. We have also tried other approaches. For example, halving only the mismatches at the reversible edges - this approach, however, did not yield satisfactory results, because then the model prefers sampling DAGs whose CPDAGs contain only very few reversible edges. This downgrades the essence of the CPDAG coupling. The approach proposed by us is in a certain sense a compromise between these two strategies and yields the best results while also aligning well with the theoretical behavior of the partition function. The hyperparameter trace plots presented in the Appendix indicate that the hyperparameters β_i are sampled in a similar fashion for each $\text{MES}^{(n,\beta_i)}$, which suggests that our proposed models for $\text{MES}^{(n,\beta_i)}_{\text{CPDAG}}$ and $\text{MES}^{(n,\beta_i)}_{\text{SKEL}}$ are a reasonable choice. There are no indications of improper behavior. In Section 5.1, we further study the dynamics of the partition function for each $\text{MES}^{(n,\beta_i)}$.

If $\mathcal{G}^*$ is a skeleton, we have $k_n = 0$ and $l_n = \sum_{i=1}^{N} \mathbb{1}\{\mathcal{G}^*_{in} = -1\}$. Given that $l_n \neq 0$, each of the l_n undirected edges can be pointing towards n or from n to another node. The "true" mismatch vector can be any of $\xi^{(0)}, \xi^{(1)}, \ldots, \xi^{\min(F,l_n)}$, respecting the fan-in restriction F. Hence, consider their mean value

$$\hat{\xi}^{(l_n)} = \frac{\sum_{j=0}^{\min(F,l_n)} \xi^{(j)}}{\min(F+1, l_n+1)}, \tag{3.25}$$

and define the $\text{MES}^{(n)}_{\text{SKEL}}$ as

$$\text{MES}^{(n,\beta)}_{\text{SKEL}} = \begin{cases} \sum_{x \in \xi^{(0)}} e^{-\beta x}, & \text{if } l_n = 0, \\ \sum_{x \in \hat{\xi}^{(l_n)}} e^{-\beta x}, & \text{if } l_n \neq 0. \end{cases} \tag{3.26}$$

To summarize, the enhanced perfect gas approximation of the partition function in each coupling scheme is

$$\begin{aligned} \text{If } \mathcal{G}^* \text{ is a DAG} &\implies Z_{\text{DAG}} = Z \approx \prod_{n=1}^{N} \text{MES}^{(n,\beta)}_{\text{DAG}} \\ \text{If } \mathcal{G}^* \text{ is a CPDAG} &\implies Z_{\text{CPDAG}} = Z \approx \prod_{n=1}^{N} \text{MES}^{(n,\beta)}_{\text{CPDAG}} \\ \text{If } \mathcal{G}^* \text{ is a skeleton} &\implies Z_{\text{SKEL}} = Z \approx \prod_{n=1}^{N} \text{MES}^{(n,\beta)}_{\text{SKEL}}. \end{aligned} \tag{3.27}$$

4 Experimental Setup

In Section 4.1, we describe how we generate datasets from a known true network, create corrupted datasets to test robustness, set up the prior and proposal distributions for the hyperparameter β, and choose parameters for computing the BGe score. Section 4.2 introduces the metrics used to assess the performance of our MCMC inference, and Section 4.3 provides additional implementation details.

4.1 Synthetic Data, Priors and other Parameters in the MCMC Scheme

We generate synthetic Raf datasets for our numerical MCMC simulations. To do so, we sample from a linear Gaussian distribution in the following way:

$$X_i \sim \mathcal{N}\left(\sum_{\mathrm{pa}_{X_k}[\mathcal{R}]} w_{i,k} X_k, \sigma^2\right). \tag{4.1}$$

Here, X_i represents the random variable - a node i in the gold-standard Raf pathway $\mathcal{R}$ of 1.1, $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with a standard deviation σ and a mean $\mu = \sum_{\mathrm{pa}_{X_k}[\mathcal{R}]} w_{ik} X_k$. We sum over the values of the parents of a node k. The terms $w_{i,k}$ are the weights that determine the interaction strengths. We choose $\sigma = 0.1$, $w_{i,k} \sim \mathcal{U}([-2, -0.5] \cup [0.5, 2])$; that is, we sample the weights uniformly from certain continuous intervals.

If we assign the random variables to the individual molecules of the Raf pathway as: $X_1 \coloneqq \mathrm{pip3}, X_2 \coloneqq \mathrm{plcg}, X_3 \coloneqq \mathrm{pip2}, X_4 \coloneqq \mathrm{pkc}, X_5 \coloneqq \mathrm{pka}, X_6 \coloneqq \mathrm{raf}, X_7 \coloneqq \mathrm{p38}, X_8 \coloneqq \mathrm{jnk}, X_9 \coloneqq \mathrm{mek}, X_{10} \coloneqq \mathrm{erk}, X_{11} \coloneqq \mathrm{akt}$, then $(X_1, X_2, \ldots, X_{11})$ is a topological ordering on $\mathcal{R}$. Thus, the synthetic data can be generated as:

$$X_1 \sim \mathcal{N}(0, \sigma^2), \tag{4.2}$$
$$X_2 \sim \mathcal{N}(w_{1,2} X_1, \sigma^2), \tag{4.3}$$
$$X_3 \sim \mathcal{N}(w_{1,3} X_1 + w_{2,3} X_2, \sigma^2), \tag{4.4}$$
$$\vdots \tag{4.5}$$
$$X_{11} \sim \mathcal{N}(w_{1,11} X_1 + w_{7,11} X_7 + w_{10,11} X_{10}, \sigma^2). \tag{4.6}$$

As it is stated in [67], an alternative way to obtain the synthetic Raf data is to use the software package Netbuilder [71]. The conceptually more correct approach to sampling the synthetic Raf data is to model the interactions of the network as enzyme-substrate reactions in organic chemistry. A set of ordinary differential equations can describe such a process [70]. The Netbuilder package provides an approximation of the analytical solution to such a set of ordinary differential equations by using sigmoidal transfer functions.

We also conduct a robust inference on the Raf data with purposefully added noisy, corrupted data. The entries in this corrupted dataset $\mathcal{D}_{\mathrm{corrupt}}$ are simply drawn from a continuous uniform distribution $\mathcal{U}[-2, 2]$. An alternative way to generate corrupted data with additive Gaussian noise is presented in [68].

Following the construction of [67], the prior distribution for the hyperparameters $\mathbb{P}(\beta^{(i)})$ is taken to be the uniform continuous interval $[0, 30]$. The proposal distribution $R(\beta^{(i)}_{\mathrm{new}} \mid \beta^{(i)}_{\mathrm{old}})$ is a uniform distribution of a moving interval of length L, centered at $\beta^{(i)}_{\mathrm{old}}$. A reflection is used so that $\beta^{(i)}_{\mathrm{new}} \in [0, \mathrm{MAX}]$. More explicitly, this works as follows.

1. Sample $\beta^{(i)}_{\mathrm{rand}} \sim \mathcal{U}[\beta^{(i)}_{\mathrm{old}} - \frac{L}{2}, \beta^{(i)}_{\mathrm{old}} + \frac{L}{2}]$.

2. Set

$$\beta^{(i)}_{\mathrm{new}} = \begin{cases} -\beta^{(i)}_{\mathrm{rand}}, & \text{if } \beta^{(i)}_{\mathrm{rand}} < 0, \\ 2\mathrm{MAX} - \beta^{(i)}_{\mathrm{rand}}, & \text{if } \beta^{(i)}_{\mathrm{rand}} > \mathrm{MAX}, \\ \beta^{(i)}_{\mathrm{rand}}, & \text{otherwise.} \end{cases} \tag{4.7}$$

In our simulations we use $L = 6$ and MAX $= 30$, though in principle, these parameters can be adjusted throughout the iterations. A sophisticated choice of gradually increasing L and MAX can result in an MCMC that resembles an annealing scheme [45].

The parameters in the computation of the Gaussian BGe score are chosen as follows: the scaling parameter $\nu = 1$; the degrees of freedom $\alpha = N + 2 = 13$; the covariance $T_0 = \frac{1}{2}\mathbf{Id}_{N\times N} = \frac{1}{2}\mathbf{Id}_{11\times 11}$; the mean $\mu_0 = \mathbf{0}_{N\times 1} = \mathbf{0}_{11\times 1}$, where $\mathbf{0}$ denotes the zero vector. Such chosen parameters correspond to a conservative setup ensuring that the data itself has the major influence, rather than the prior assumptions [27, 23].

4.2 Evaluation Metrics

As we emphasize many times in this paper, not all of the edge directions in a Bayesian network can always be inferred, in practice we can only learn the equivalence classes of the Bayesian network, which are encoded in its CPDAG. Therefore, in our MCMC scheme, we sample networks as CPDAGs. As it is common in Bayesian inference, at the end of the sampling process, the resulting sampled networks corresponding to the distinct datasets $\mathcal{D}_1, \ldots, \mathcal{D}_I$ are averaged. This averaging, assuming that we run the MCMC until full convergence, corresponds to the mean of the corresponding posterior probability distribution $\mathbb{P}(\mathcal{G}^{(i)} \mid \mathcal{D}_i)$ and can be used for inference. We thus obtain I averaged CPDAG networks. We compare each of the I averaged CPDAGs to the Raf CPDAG, which serves as the ground CPDAG truth. Additionally, we also compare the underlying averaged I skeletons to the Raf skeleton, which serves as the ground skeleton truth. To assess the performance of our method(s), we use two, in computational statistics and machine learning, commonly used evaluation metrics - the Receiver Operator Curve (ROC) and the Precision-Recall (PR) [14].

The ROC curve plots the True Positive Rate (TPR), also called sensitivity or recall, against the False Positive Rate (FPR) at various classification thresholds.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.8}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{4.9}$$

TP, true positives, denotes the number of correctly classified positive classes, that is, instances when we correctly learn the edge $X \to Y$. TN, true negatives, denotes the number of times when we correctly do not assign the edge $X \to Y$. FN, false negatives, denotes the number of times when we miss learning an edge that is present in the ground truth network. FP, false positives, denotes the number of times when we learn an edge that is not present in the ground truth network.

The PR plots Precision against Recall (TPR) at different classification thresholds. We have

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4.10}$$

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{FP} + \text{TN}}. \tag{4.11}$$

The ROC is especially an accurate performance metric if we have balanced datasets when the fraction of positive and negative classes are close to uniform. Even though ROC provides a general understanding of model performance, it can be deceptive if the fraction of positive and negative classes is far from being uniform. The maximum number of nodes in a DAG on N nodes is $\binom{N}{2} = \frac{N(N-1)}{2}$. The gold-standard Raf has 19 edges in its DAG, 35 edges in its CPDAG, and 38 edges in its skeleton[†]. The maximum number of edges of a DAG on $N = 11$ nodes is 55, while for CPDAGs and skeletons, this can be 110. This looks rather unbalanced, and thus, mindlessly using only the ROC metric would not be appropriate. PR curves provide a more accurate picture of the performance of our methods with such skewed network structures.

To get the best of the two worlds, we use both ROC and PR and calculate the Area Under Curve (AUC). AUC is a real number from the interval $[0, 1]$. Larger AUC values indicate a better prediction performance overall, and AUC $= 1$ corresponds to the ideal model. For ROC, another important indicator of the model performance is the slope of the resulting curve at values near the origin.

[†]Assuming that we interpret reversible edges as bidirected ones.

Steep slopes mean we observe a high TPR and low FPR, which means that the model is increasing true positives significantly without increasing false positives much, which is desirable. Shallow slopes indicate poor discrimination. In PR, desirable curves are those that stay in the top right region, meaning that both the precision and recall are high across most thresholds [14]. Figures 5.5 and 5.6 show examples of ROC AUC and PR AUC plots obtained in our simulations.

For ROC, AUC below the baseline value of 0.5 indicates the poor performance of the model, as this baseline value corresponds to random guessing. ROC does not depend on class distribution because it considers both positive and negative samples through TPR and FPR, which are normalized. For PR curves, the baseline AUC value that corresponds to random guessing is equal to the fraction of positive classes in the dataset. For the gold-standard Raf, this is equal to 0.31 in the CPDAG case and 0.35 in the skeleton case.

4.3 Software Implementation

Beyond the theoretical contributions of Section 3, we have developed a complete, from-scratch software implementation in `Python`. Owing to its substantial size, the full source code for the coupled model(s) is hosted on a GitHub repository[†]. The repository contains the following scripts: `model.py`, `config.py`, `Partition_function_empirical_analysis.ipynb`, `StatisticalAnalysis.ipynb`, and `data.py`.

`model.py`
This script encapsulates most of the algorithms described in this project. In particular, it contains:

1. A function to compute a topological ordering. (Originally provided as an `R` script in the Statistical Genomics course (WMMA008-05), it was translated to `Python` with the help of generative AI [52] and additional debugging.)
2. The DAG-to-CPDAG algorithm based on [11]. (Originally provided as an `R` script in the Statistical Genomics course (WMMA008-05), it was translated to `Python` with the help of generative AI [52] and additional debugging.)
3. The BGe score calculation. (Originally provided as an `R` script in the Statistical Genomics course (WMMA008-05), it was translated to `Python` with the help of generative AI [52] and additional debugging.)
4. Data generation routines as described in Section 4.1.
5. Coupling mechanisms for DAGs, CPDAGs, and skeletons.
6. Implementations of Algorithms 3.1, 3.2, and 3.3.
7. The Gibbs model 2.30.
8. Functions to generate neighbor DAGs using edge additions, removals, and reversals (see Section 2.2).
9. The MCMC scheme from Section 2.5.2, including all proposal distributions.
10. Evaluation metrics (ROC and PR) from Section 4.2, plus additional visualization functions for the figures in Section 5, and heatmap networks showing posterior edge probabilities.
11. A launcher function designed for easy modification to support parallel computing.
12. Other helper functions and meta-optimization schemes, such as "dictionary compression" to accelerate partition function computations when a proposed hypernetwork/hyperparameter β_i is not accepted in the MCMC algorithm.

`config.py`
A configuration file that contains all input networks and parameters for `model.py` in one place, enabling flexible input adjustments and control over the simulations.

`Partition_function_empirical_analysis.ipynb`
A Jupyter notebook containing the code used in Section 5.1. This notebook measures the CPU time

[†]https://github.com/RadovanRusnak/Master_Project_Implementation

required by PGA and EPGA to compute partition functions for randomly generated DAGs of varying size. It also includes code to visualize the approximated logarithm of the partition function for each MES (see Section 3.4).

`StatisticalAnalysis.ipynb`
This notebook performs the statistical analyses described in Section 5.2. Specifically, it implements the Shapiro–Wilk test for normality, pairwise t-tests with confidence intervals to assess statistical significance, and a one-way ANOVA.

`data.py`
Running this script generates a `JSON` file that stores all AUC results from the simulation studies. These datasets are then used for the statistical analysis.

The large set of simulations (Sections 5.2 and 5.3) was run on a remote High Performance Computing (HPC) cluster, Habrók, which is operated by the University of Groningen. Habrók is designed to handle large and complex computational tasks, making it ideal for computationally intensive tasks such as MCMC sampling.

5 Results

Section 5.1 presents the figures resulting from an empirical analysis of the partition functions in the Gibbs model. We begin by comparing the time complexity of PGA and EPGA on randomly generated DAGs of varying sizes. Next, we visualize the behavior of the partition function under each of the three coupling schemes: DAG, CPDAG, and skeleton coupling. Section 5.2 highlights the key findings from our simulation studies. We report the results of our MCMC simulations and evaluate the significance of each coupling scheme relative to the uncoupled Bayesian model using pairwise t-tests. In addition, we conduct a one-way ANOVA test to assess whether the proposed CPDAG and skeleton couplings lead to improvements in ROC and PR AUC, specifically for Raf CPDAG and Raf skeleton reconstruction. Section 5.3 summarizes our findings on the robustness of the Bayesian model. Here, we intentionally include corrupted datasets alongside the true data to evaluate the model's performance under such conditions. Finally, Section 5.4 presents our findings regarding the convergence of the MCMC iterations.

5.1 Empirical Assessment of the Computation and Behavior of the Partition Functions

To evaluate the effectiveness of EPGA 3.19 in comparison to PGA 3.8, we conduct a series of empirical experiments. Specifically, we generate 20 random DAGs with an increasing number of nodes, ranging from $N = 5$ to $N = 100$, ensuring that each adheres to the fan-in restriction $F = 3$. For both methods, we measure the CPU time per DAG required for successful termination, expressed in seconds. Additionally, we compute their cumulative CPU time and plot the speedup ratio, defined as the ratio PGA/EPGA of the individual CPU times per DAG. The resulting plots are presented in Figure 5.1. This empirical experiment suggests that EPGA achieves a significant performance improvement, successfully mitigating the computational bottleneck inherent in PGA. This will allow us in future experiments to further increase the number of MCMC steps and thus hopefully improve the precision of the sampling.

Next, let us briefly perform an empirical analysis of the behavior of each of the partition functions as proposed in Section 3.4. We plot for $\beta \in (0, 10]$, in the same figure, the logarithm of the quantities $Z_{\text{DAG}}, Z_{\text{CPDAG}}$ and Z_{SKEL} using the DAG, CPDAG, and skeleton structures of the gold-standard Raf pathways, respectively. The resulting plot can be seen in Figure 5.2. While the 3 curves might seem almost identical, the dip-hump structure in Figure 5.3 captures the numerical differences between each partition function. Various factors contribute to the numerical behaviour of $Z_{\text{DAG}}, Z_{\text{CPDAG}}$ and Z_{SKEL}. In particular, the control coupling hyperparameter β is sampled within the MCMC scheme from Section 2.5.2, and its dynamics further affect the behavior of each partition function. To further illustrate the behavior and differences between $Z_{\text{DAG}}, Z_{\text{CPDAG}}$ and Z_{SKEL}, we generate 20 random DAGs of various sizes, satisfying the fan-in restriction $F = 3$, and plot the mean differences between each partition function as a function of $\beta \in (0, 10]$. The results can be seen in Figure 5.4. Notice that the dip-hump structure between the partition functions becomes more pronounced with the growing size of the networks. The most obvious explanation of this empirical experiment is that in larger networks, the magnitude range of the mismatch space increases, which further amplifies or mitigates the corresponding $\text{MES}^{(n,\beta)}$. This observation is rather counterintuitive and will therefore be further discussed in Section 6.1.

5.2 Raf Reconstruction: The Efficiency of the Coupled Methods

For the simulations in this section, we generate three groups of datasets with varying sizes to ensure an unbiased analysis. In the first group, we form 9 chunks of datasets $\mathcal{D}_1, \ldots, \mathcal{D}_5$, where each dataset has $|\mathcal{D}_i| = 10$ observations. In the second group, we form 9 chunks where each dataset has $|\mathcal{D}_i| = 50$ observations. In the third group, we form 9 chunks with each dataset containing $|\mathcal{D}_i| = 100$ observations. In all cases, each chunk is generated using a different random probability seed, specifically using seeds $35, 36, \ldots, 43$. Next, we test 4 methods: CPDAG coupling (Section 3.1), Skeleton coupling (Section 3.2), DAG coupling (Section 2.5), and a no-coupling method. In total, we run $4 \times 3 \times 9 = 108$ MCMC simulations. The networks $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(5)}, \mathcal{G}^*$ in the MCMC scheme can be initialized as empty graphs or, as suggested in [67], pre-trained before sampling. In our approach, we initialize the MCMC scheme by setting $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(5)}, \mathcal{G}^*$ to their consensus network, thereby providing a headstart that saves time. We perform several rounds of empirical experiments, average the resulting networks, and retain the edges that are consistently identified across experiments. The resulting consensus network is used to initialize the networks in the main MCMC sampling.

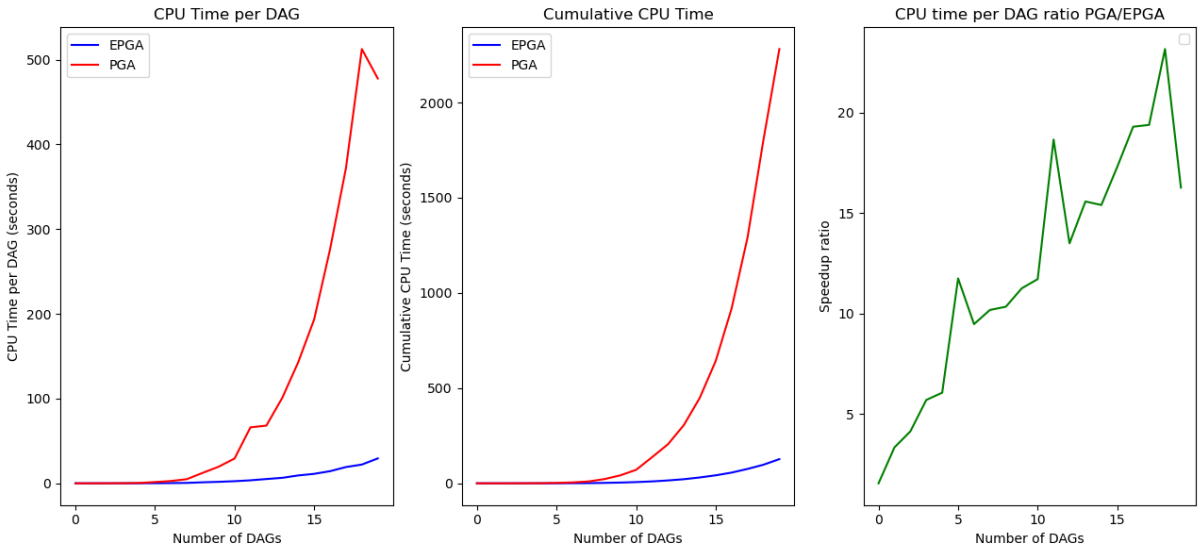


Figure 5.1: Comparison of EPGA and PGA via measuring the individual and cumulative CPU time needed for termination. The x-axis goes through the 20 randomly generated DAGs with an increasing number of nodes $N = 5, 10, \ldots, 100$. We impose a fan-in restriction $F = 3$.

We can turn off the coupling by setting the hypernetwork $\mathcal{G}^*$ to the same value as the network $\mathcal{G}^{(i)}$, for $i = 1, \ldots, I$ in the MCMC algorithm. This ensures that the prior probability for the network $\mathcal{G}^{(i)}$ is given by $\mathbb{P}(\mathcal{G}^{(i)} \mid \beta^{(i)}) = \frac{1}{Z(\beta^{(i)}, \mathcal{G}^{(i)})}$. Thus, we define the no-coupling method by setting $\beta^{(i)} := 1$ and using the DAG partition function $Z := Z_{\text{DAG}}$.

All MCMC simulations are run with 10^6 steps. The burn-in phase is set to 50%, meaning that we start saving the generated samples only after completing 5×10^5 MCMC iterations. The inclusion of a burn-in phase is a widely used strategy to reduce the initial bias introduced by the initialization and to avoid sampling from low-probability regions that are far from the target posterior distribution. We do not apply any thinning to the MCMC chains, as thinning can lead to potential precision and information loss. Since the experiments are conducted on the HPC Habrók, we are not constrained by RAM or CPU time. Under such conditions, thinning is unnecessary for accurate posterior modelling [40, 59].

Specifically, we aim to assess the impact of the coupling on network performance. To evaluate this effect, we compute the network-wise differences in ROC and PR AUC between the averaged networks obtained using each coupling method and those generated using the uncoupled method. This evaluation is conducted for each data size $|\mathcal{D}_i| \in \{10, 50, 100\}$. Conducting the Shapiro-Wilk test verifies the normality of the obtained network-wise AUC differences. Subsequently, we perform pairwise t-tests to calculate the 95% Confidence Intervals (CIs), which allow us to test the hypothesis that the network-wise AUC differences between the coupled and uncoupled methods are statistically significant. The results, including the computed CIs and the corresponding mean network-wise AUC differences for each coupling method and data size, are presented in Figures 5.8, 5.9, 5.11, and 5.10. The visualized CIs reveal several important insights.

First, the coupling effect is most pronounced with the smallest data size, gradually diminishing as the data size increases. This observation aligns with expectations, as information sharing via the hypernetwork $\mathcal{G}^*$ facilitates the transfer of knowledge between different datasets, a benefit not observed when learning the networks $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(5)}$ separately. As the amount of data increases, the necessity for coupling decreases since each dataset contains a sufficient amount of information, allowing for effective learning of the networks individually.

Second, both the CPDAG and Skeleton couplings outperform the uncoupled approach. According to pairwise t-tests, this outperformance is statistically significant in all evaluation metrics and data sizes. This can be concluded by the observation that each CI lies above the horizontal line $y = 0$. In

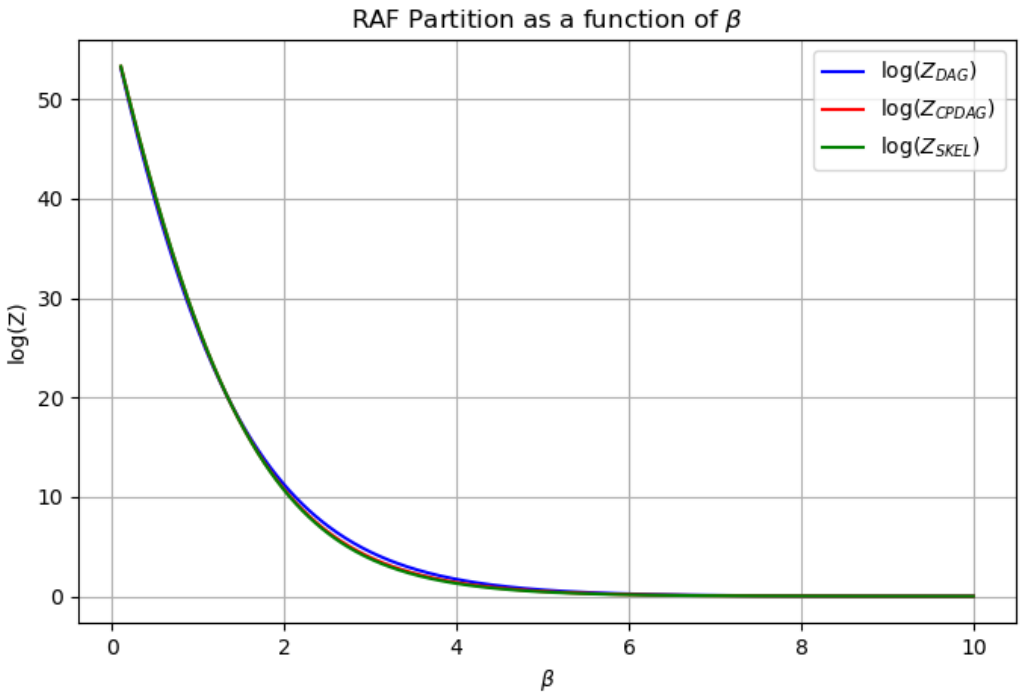


Figure 5.2: Logarithm of the partition functions $Z_{\text{DAG}}, Z_{\text{CPDAG}}$ and Z_{SKEL} using the gold-standard Raf pathway across multiple values of β. The partition functions are nonlinear and monotonically decreasing.

contrast, the DAG coupling does not consistently show such superiority, particularly when $|\mathcal{D}_i| = 100$. Furthermore, the figures 5.8, 5.9, 5.11, and 5.10 suggest that the proposed CPDAG and Skeleton coupling schemes might be more efficient.

To formally assess whether there is a statistically significant difference between the coupled methods, we perform a one-way ANOVA test, with each coupling method as the single main effect. The AUC differences are considered across all four evaluation metrics: Skeleton ROC AUC, CPDAG ROC AUC, Skeleton PR AUC, and CPDAG PR AUC. The ANOVA yields an extremely small p-value of 2.16×10^{-16} and F-statistic $F(2, 1617) = 34.48$. This allows us to reject the null hypothesis that all methods perform equivalently.

Next, we conduct a post-hoc analysis using pairwise t-tests to determine which specific pairs of methods indicate statistically significant differences. Since multiple pairwise comparisons are performed, we apply the Bonferroni correction to adjust the significance threshold for the p-values. Comparing the CPDAG and Skeleton approaches, we obtain a p-value of 0.607, which is greater than the adjusted p-value threshold $\frac{0.05}{3} \approx 0.0167$, indicating no significant difference between these two methods in terms of their network-wise AUC differences. However, comparing the CPDAG and DAG approaches results in a p-value of 1.69×10^{-13}, indicating a statistically significant difference. A similar result is observed when comparing the Skeleton and DAG approaches, yielding a p-value of 2.91×10^{-12}, which is also less than 0.0167. These statistical findings suggest that the proposed CPDAG and Skeleton coupling schemes, as detailed in Sections 3.1 and 3.2, provide a further improvement over the original DAG coupling method proposed in [67].

In our simulations, we occasionally observe instances where some of the networks fail to couple. This typically results in a noticeable performance drop in terms of the AUC for the corresponding network. To illustrate this situation, consider the ROC AUC results presented in Figure 5.5 and the PR AUC results shown in Figure 5.6. The performance decline can be observed in the network associated with the dataset $\mathcal{D}_1$. Figure 5.7 displays the histograms of the sampled hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$. It is evident that $\beta^{(1)}$ is centered around a smaller value compared to the other hyperparameters, which may

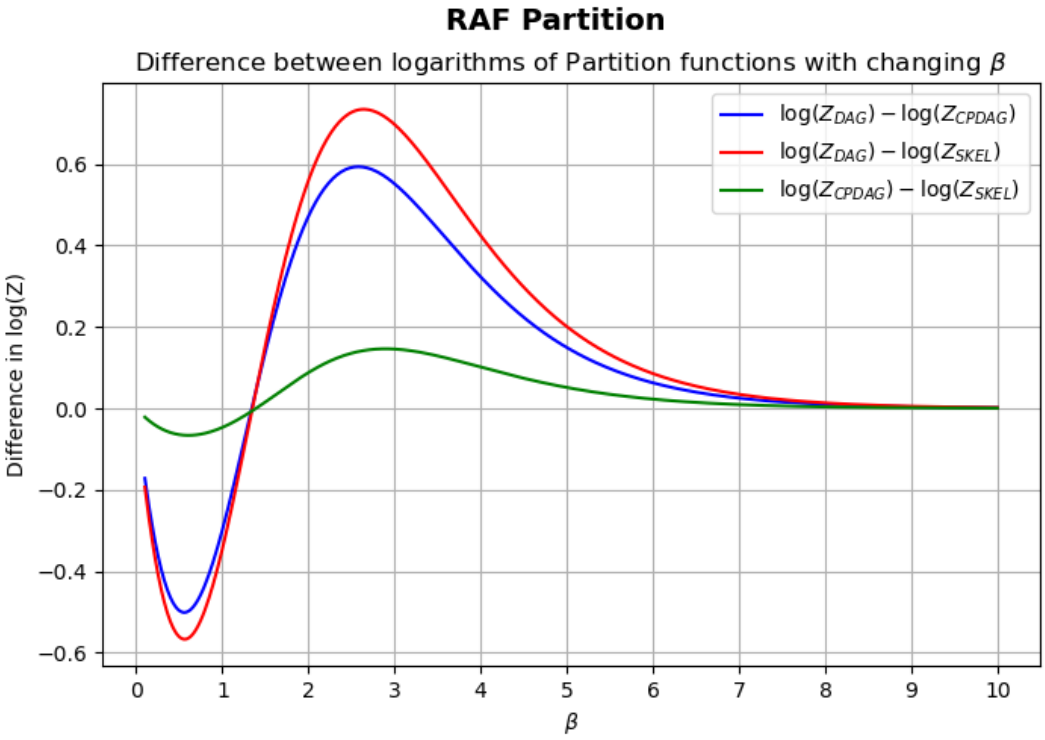


Figure 5.3: Difference between the logarithms of $Z_{\text{DAG}}, Z_{\text{CPDAG}}$ and Z_{SKEL} with respect to the gold-standard Raf pathway and changing β's. Notice the fluctuations shown by the difference curves.

explain the observed performance degradation.

5.3 Robust Inference: Inclusion of Corrupted Data

This section examines whether the coupled models can effectively differentiate between true Raf data and corrupted data consisting of noise. Corrupted data can in practice correspond to, for example, a failed biological experiment. Since the hyperparameter $\beta^{(i)}$ governs the coupling strength of the network $\mathcal{G}^{(i)}$ with other networks, the detection of corrupted data can be indicated if the MCMC scheme samples the corresponding hyperparameters around small values. To investigate this, we generate 4 true Raf datasets from the gold-standard Raf network using 4.1, while the fifth dataset, $\mathcal{D}_5 = \mathcal{D}_{\text{corrupt}}$, consists of noisy data as described in Section 4.1. Our primary interest lies in the sampled hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$, specifically their trace plots obtained from the MCMC scheme. The expected and ideal outcome is that the hyperparameters associated with the true datasets are sampled uniformly across the interval $[0, 30]$, whereas the hyperparameter $\beta^{(5)}$, corresponding to the noisy dataset, remains low. At 3 different probability seeds, we generate 3 groups of datasets $(\mathcal{D}_1, \ldots, \mathcal{D}_5)$, each group varying in data size, with $|\mathcal{D}_i| \in \{10, 50, 100\}$. We evaluate 3 coupling methods: CPDAG coupling (Section 3.1), Skeleton coupling (Section 3.2), and DAG coupling (Section 2.5). This results in a total of $3 \times 3 \times 3 = 27$ MCMC simulations.

Additionally, we aim to determine whether the newly introduced CPDAG and skeleton coupling schemes offer improvements over the original DAG coupling approach. Unlike in previous sections, all networks in this simulation study are initialized as empty graphs. This choice ensures a fair evaluation of the effectiveness of each method concerning the primary objective. All MCMC simulations are conducted with 10^6 steps and a 50% burn-in phase.

We present the results in various figures, each consisting of three rows, where each row corresponds to a different random probability seed. Figures .1, .2, and .3 illustrate the CPDAG coupling results, figures .4, .5, and .6 correspond to the skeleton coupling results, and figures .7, .8, and .9 depict the DAG coupling results. The figures are shown in the Appendix due to their large size. The results are

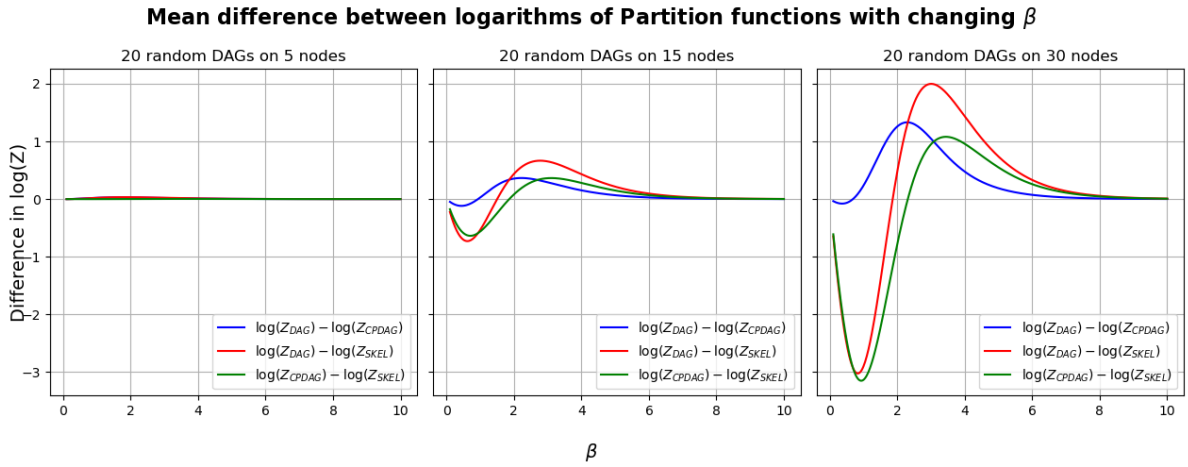


Figure 5.4: Similar to the Figure 5.3, but the input networks are randomly generated DAGs on $N = 5, 15, 30$ nodes. The mean difference between the logarithms of $Z_{\mathrm{DAG}}, Z_{\mathrm{CPDAG}}$ and Z_{SKEL} with respect to the changing β's is plotted. Notice that the differences between the partition functions become more significant with larger sizes of the DAGs.

somewhat unsatisfactory. Although, in most cases, the hyperparameter $\beta^{(5)}$ is sampled consistently at low values, a major issue arises. Occasionally, we observe an inverted coupling scenario—contrary to expectations—where the corrupted data network $\mathcal{G}^{(5)}$ exhibits strong coupling, while the coupling of the other networks is suppressed. This phenomenon causes the hyperparameter $\beta^{(5)}$ to fluctuate significantly across the range $[0, 30]$. As a consequence, the other hyperparameters, $\beta^{(1)}, \ldots, \beta^{(4)}$, can either "catch and poison" themselves due to this coupling, forcing the networks $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(4)}$ to resemble $\mathcal{G}^{(5)}$, or the coupling is suppressed, and the networks $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(4)}$ are effectively learned as separate networks.

5.4 Convergence of the MCMC Scheme

Unfortunately, convergence issues were present in every large-scale simulation conducted throughout this project, including both the experiments from Sections 5.2 and 5.3. Similar challenges have been reported in [67] despite that the authors use only 5×10^5 MCMC steps. Even with the EPGA approximation scheme developed in Section 3.3, which enables us to allocate additional MCMC iterations by optimizing CPU time, we were unable to resolve these convergence difficulties.

The scatter plot in Figure 5.12 illustrates the marginal edge posterior probabilities obtained from two independent MCMC realizations under one of our coupled schemes. Ideally, if proper convergence were achieved, all marginal posterior probabilities would align along the diagonal. However, the observed deviations suggest that our simulations did not fully exploit the capabilities of the Bayesian model. This highlights the potential for further improvement, particularly through the adoption of more sophisticated sampling techniques beyond the classical Metropolis-Hastings algorithm, which often struggles with multi-modal distributions, leading to chains becoming trapped in local modes [13].

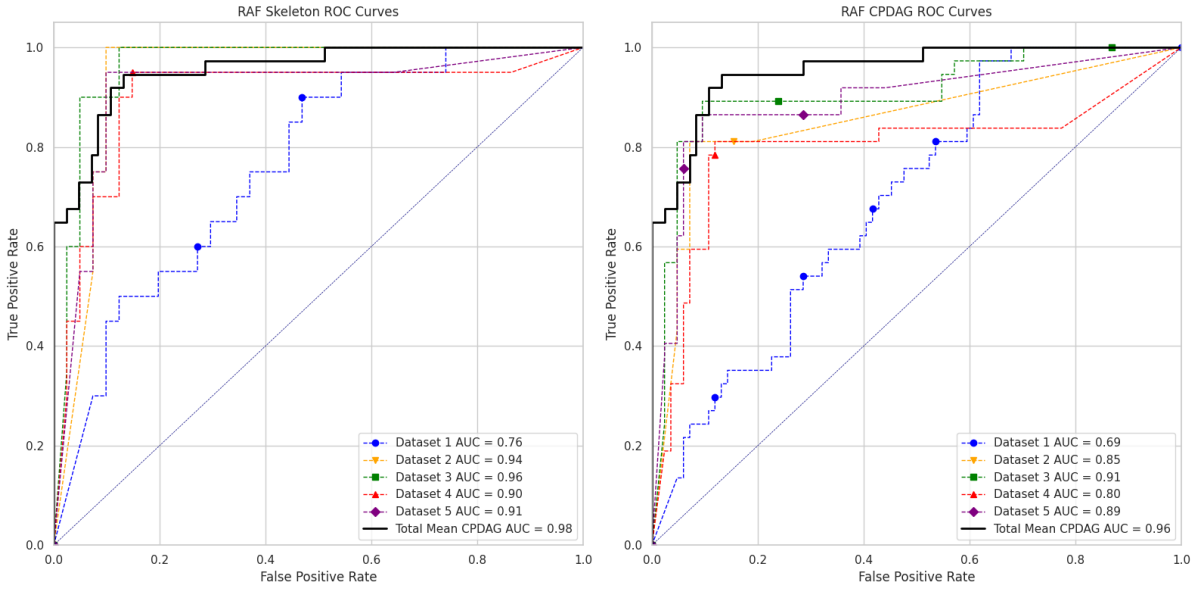


Figure 5.5: The picture shows the ROC curves and the AUC values of 5 averaged CPDAG networks sampled by the MCMC scheme. The black curve represents a network created by averaging the 5 already averaged CPDAG networks. On the left, we evaluate the ROC AUC, comparing it to the Raf skeleton. On the right, we evaluate the ROC AUC, comparing it to the Raf CPDAG. Notice the steep slopes on the left sides of each picture, which is an indication of a good model performance. Also, notice that the learned network corresponding to the dataset $\mathcal{D}_1$ shows a poorer performance than the other networks. This can be explained by the fact that the network did not couple properly throughout the sampling process, as can be seen in Figure 5.7.

6 Discussion and Conclusion

6.1 Key Theoretical Contributions

This thesis is written in a self-contained manner. We introduce the Bayesian model based on DAG coupling interactions over the Gibbs prior from [67]. We then extend this model by considering both the equivalence classes of DAGs—resulting in CPDAG coupling using the SHD metric—and coupling DAGs based purely on mismatches between their underlying skeleton structures. The main motivation behind these coupling schemes is to avoid penalizing edge mismatches in equivalent DAG networks. Using the score equivalence metric BGe, one can realistically infer only the CPDAG when attempting to reverse-engineer a ground truth network, such as the Raf pathway. Special attention is given to studying the size and number of mismatches that can occur between parent configurations in DAG networks and the proposed parent configurations subject to a fan-in restriction. The result is the algorithm SAKI, which allows us to precompute a matrix containing all feasible mismatch scenarios needed to compute the partition function of the Gibbs distribution.

Most importantly, SAKI enables us to define the EPGA approximation algorithm, which, according to Figure 5.1, demonstrably outperforms the sometimes computationally cumbersome PGA. One may protest that we should generate several DAGs for each $N \in \{5, 10, \ldots, 100\}$ and work, for each N, with the averages instead. However, this is not necessary for an unbiased experiment. The computation of the partition function Z using both EPGA and PGA is purely deterministic in the sense that, for a fixed N and F, we always perform the same number of numeric operations for a given approximation method. Recall from Section 3.3 that the cardinality of π_n and the number of columns of the matrix Ξ from 3.18 is both equal to $R = \sum_{f=0}^{F} \binom{N-1}{f}$. In EPGA 3.19, we perform the R additions $e^{-\beta x_1} + \cdots + e^{-\beta x_R}$ exactly N-times. In contrast, PGA requires performing N-times R additions of terms $e^{-\beta \mathcal{E}(\cdot,\cdot)}$, where in addition each $\mathcal{E}(\cdot,\cdot)$ requires additional $N^{\dagger}$ runs through the nodes of the DAG. Consequently, the speedup ratio should grow proportionally to the

[†]Or $N-1$ if we exclude the diagonal terms in the adjacency matrix of a given DAG, as DAGs do not allow self-loops.

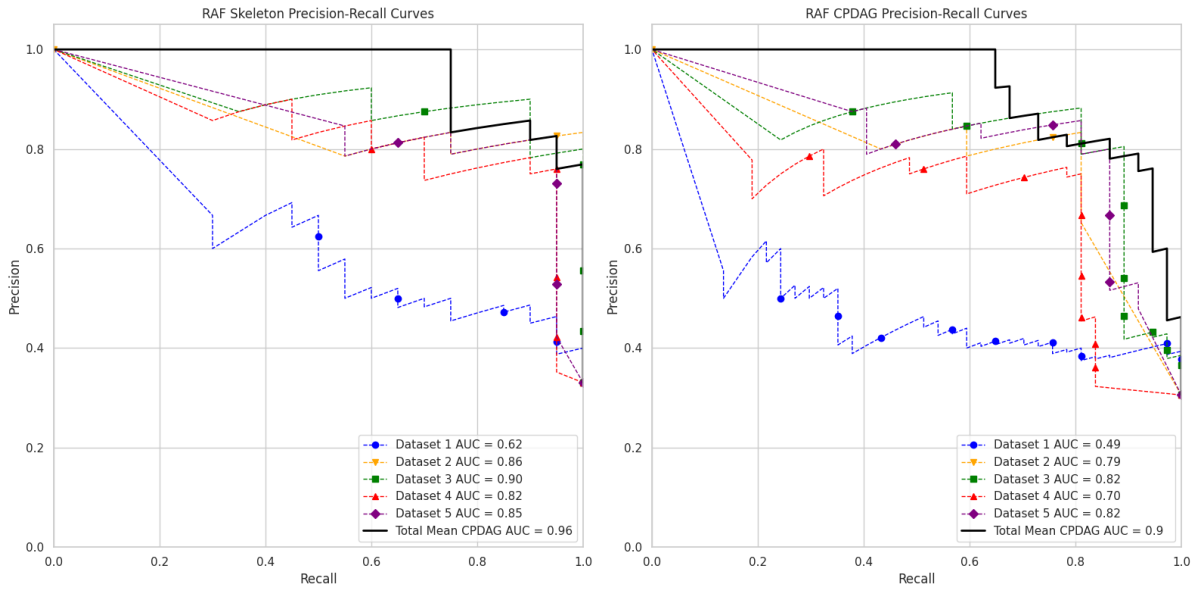


Figure 5.6: The picture shows the PR curves and the AUC values of 5 averaged CPDAG networks sampled by the MCMC scheme. The black curve represents a network created by averaging the 5 already averaged CPDAG networks. On the left, we evaluate the PR AUC, comparing it to the Raf skeleton. On the right, we evaluate the PR AUC, comparing it to the Raf CPDAG. Notice that the learned network corresponding to the dataset $\mathcal{D}_1$ shows a poorer performance than the other networks. This can be explained by the fact that the network did not couple properly throughout the sampling process, as can be seen in Figure 5.7.

number of nodes in the DAG. This is practically confirmed by the first-from-the-right plot in Figure 5.1. The fluctuations are due to the additional optimization meta-steps in the implementation of the EPGA.

The final theoretical contribution, incorporating some heuristics, is the introduction of MES formulas, which allow us to handle the computation of the partition function when the input networks are CPDAGs or skeletons. One slightly counterintuitive finding suggests that our prior expectations regarding CPDAG and skeleton couplings may have been somewhat overoptimistic. Since a given skeleton can encode multiple different CPDAGs and an even greater number of DAGs, intuition tempts us always to expect $Z_{\text{DAG}} \geq Z_{\text{CPDAG}} \geq Z_{\text{SKEL}}$. However, the greater network flexibility of a CPDAG or skeleton also shapes the space of mismatches between the network and proposed parent configurations - stored in the matrix Ξ from 3.18. While allowing networks greater flexibility in their edge structures—by permitting both directed and undirected edge orientations—captures a broader range of patterns in the Raf pathway, it also increases uncertainty in individual nodes' parent configurations. Consequently, the space of mismatches can expand or contract more rapidly, influencing the partition function computed by EPGA.

Assume we have a DAG $\mathcal{G}^*$ and consider the situation when one of its nodes X has one parent and X itself is a parent to 2 other nodes in $\mathcal{G}^*$. If, after converting $\mathcal{G}^*$ to a CPDAG, the edges adjacent to X are all reversible, in the CPDAG case we obtain $k_n = 0$, $l_n = 3$, and $\hat{\xi}^{(0,3)} = \frac{\sum_{j=0}^{3} \xi^{(j)}}{4}$. If using $N = 11$ and $F = 3$, we would get $\|\hat{\xi}^{(0,3)}\|_{L_1} = 586 > 544 = \|\xi^{(1)}\|_{L_1}$. One of the reasons for working with CPDAG/skeleton coupling instead of DAG coupling was that measuring the SHD between equivalence classes (or the standard Hamming distance between the underlying skeletons) should nullify unnecessary penalization of mismatches between statistically indistinguishable edges. But what we obtain in this example is even more mismatches! While at first, that might seem counterintuitive, it is the expected behavior because reversible edges can point in both directions, and consequently, we have to take this into account in our model. The opposite situation arises if the node X in the DAG $\mathcal{G}^*$ had, say, 3 parents while in the corresponding CPDAG, we would observe those edges as reversible. Then we would obtain $\|\hat{\xi}^{(0,3)}\|_{L_1} = 586 < 712 = \|\xi^{(3)}\|_{L_1}$. In this case, we observe fewer mismatches in the CPDAG in comparison with the DAG. Thus, the encoded independencies and the degree of connectedness of the underlying DAG network determines whether $\text{MES}^{(n,\beta)}_{\text{DAG}} < \text{MES}^{(n,\beta)}_{\text{CPDAG}}$ or $\text{MES}^{(n,\beta)}_{\text{DAG}} > \text{MES}^{(n,\beta)}_{\text{CPDAG}}$, which

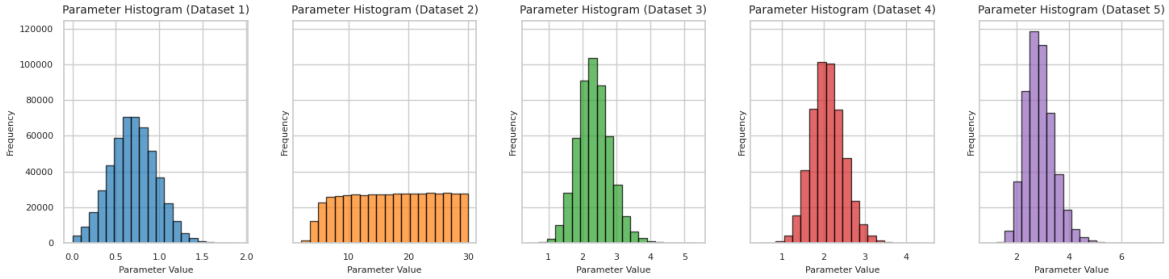


Figure 5.7: Each histogram shows the modelled posterior distribution of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ associated with the networks $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(5)}$. Larger values of $\beta^{(I)}$ indicate a stronger coupling, while lower values indicate a weaker coupling. Notice that the peak of the histogram showing $\beta^{(1)}$ is centered around the value 0.7, while the peaks of the other histograms are centered around higher values or are diffusely spread across $[0, 30]$. This suggests that the network $\mathcal{G}^{(1)}$ did not couple properly with the hypernetwork $\mathcal{G}^*$ (and thus also with $\mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(5)}$), which might explain the poor ROC and PR AUC performance shown in Figures 5.5 and 5.6.

determines the value of the resulting partition functions.

6.2 Key Experimental Findings

The most important result of this project is that the proposed coupling mechanisms outperform the model in which coupling was disabled. According to the pairwise t-test we conducted, this outperformance is statistically significant. The 95% CIs presented in Section 5.2 suggest that the coupled method exhibits the greatest superiority when data is most limited. This is particularly important in biological and medical fields, where conducting unnecessary experiments can be costly. Furthermore, the one-way ANOVA and post-hoc analysis indicate that the CPDAG and skeleton couplings outperform the uncoupled approach significantly more than the original DAG coupling does. In fact, in some cases, the DAG coupling struggles with its performance.

On the other hand, the robust inference analysis yielded less satisfactory results. The hyperparameter trace plots shown in the Appendix reveal several issues within the coupling of networks associated with the true Raf data and the corrupted data. Among the tested methods, DAG coupling visually appears to be the least effective, but we do not conduct any rigorous statistical analysis for these simulations. Major challenge observed in all three coupling approaches emerges when the dataset size is small, specifically when $|\mathcal{D}_i| = 10$. In such cases, we observe that the averaged sampled network for $\mathcal{G}^{(5)}$ closely resembles the other averaged networks. This suggests that small datasets do not provide sufficient information to extract meaningful structural insights. However, this issue diminishes as the dataset size increases. Surprisingly, despite the hyperparameter trace plots not aligning with expectations, the averaged networks obtained at the output of the MCMC sampling frequently retain characteristics indicative of the true origin of the data. There appear to be three additional issues with our simulations. First, many MCMC simulations did not run until full convergence. Second, the initialization of the MCMC sampling with empty graphs is problematic. Such initialization gives an advantage to the hyperparameter $\beta^{(5)}$ related to the corrupted data: networks inferred from the corrupted data tend to score highly but only contain a few edges, as there are no real associations between the randomized nodes. This results in these networks being similar to the hypernetwork $\mathcal{G}^*$—also initialized as an empty graph. This helps explain the high values of $\beta^{(5)}$ in some of our simulations [67]. Lastly, the intrinsic variance present within the synthetic data might prevent us from fully exploiting the data without further increasing the number of MCMC iterations. This represents an additional computational burden.

6.3 Critique and Future Research

An indisputably valid critique concerns the definition of the formulas $\text{MES}^{(n,\beta)}_{\text{CPDAG}}$ and $\text{MES}^{(n,\beta)}_{\text{SKEL}}$ in Section 3.4. Consider, for instance, the CPDAG case, where a node n has l_n undirected edges and k_n parents. In $\text{MES}^{(n,\beta)}_{\text{CPDAG}}$, we assume that any number from $\{k_n, \ldots, \min(F + 1 - k_n, l_n + 1)\}$

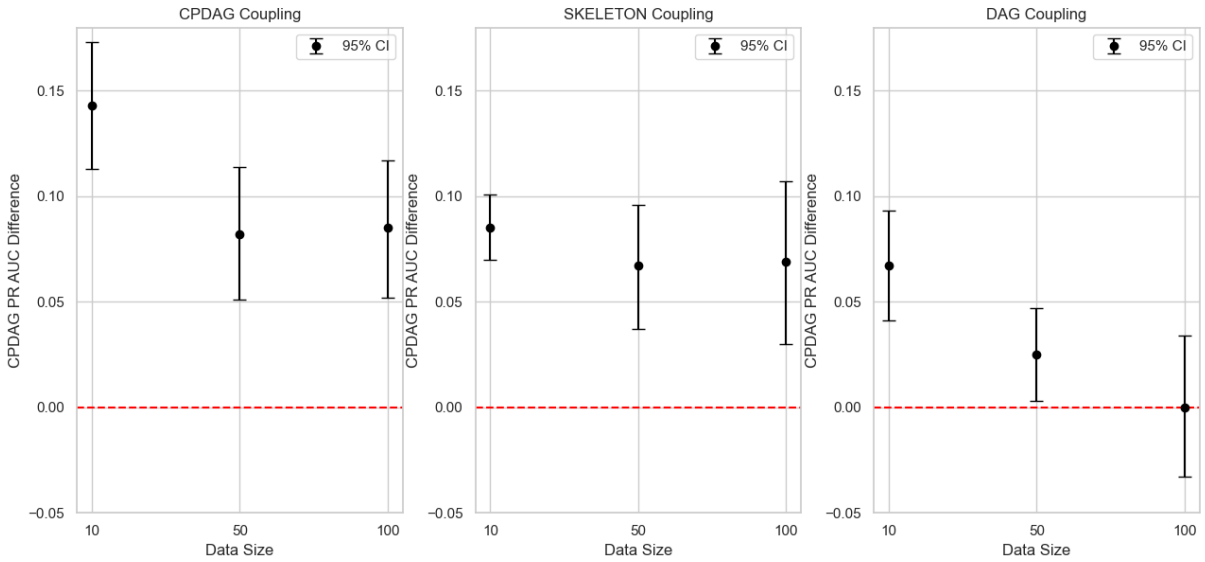


Figure 5.8: The figure shows the 95% Confidence Intervals for the network-wise differences between the coupled methods and the uncoupled method. The evaluation metric is the Raf CPDAG PR AUC.

is equally likely to be the potentially new parent configuration of n in the associated network. However, what if assigning directions to some of the l_n undirected edges introduces a new v-structure within the network? It is unclear how to exclude such potential v-structures from the formula while minimizing additional complexity. A possible alternative is to extend the SAKI algorithm to handle different types of mismatches—for instance, by defining the (local) SHD similarly to how the (local) Hamming distance was described by $\mathcal{E}(\cdot, \cdot)$ in 3.10. This remains an open problem for future research.

Perhaps the major issue we encountered was the lack of convergence of the MCMC. In theory, the convergence of MCMC samples to the posterior distribution is guaranteed, so an obvious brute-force solution would be to further increase the number of MCMC steps. More sophisticated and perhaps more promising alternatives include annealing-based approaches, such as Population MCMC [9] or Simulated annealing [4]. These methods allow for a more effective exploration of the parameter space by smoothing the energy landscape and improving mixing. Moreover, the parallel computing structure of annealing schemes offers an additional advantage: whereas classical MCMC methods suffer from interdependencies that hinder efficient parallelization [16], annealing-based techniques can leverage parallel computing to increase the number of sampling steps. The inclusion of the EPGA algorithm can provide an additional boost. Lastly, the results of our experiments suggest that initializing the MCMC with an empty graph may prolong the convergence period. Thus, providing a certain headstart to the networks by, for example, pre-training them might be beneficial. This can be achieved through separate pre-training optimization algorithms or by gradually adjusting the moving interval L and the domain interval $[0, \text{MAX}]$ of the hyperparameters $\beta^{(i)}$ throughout the MCMC iterations. In other words, we would initially learn the networks separately and gradually increase the weight of the coupling with more iterations.

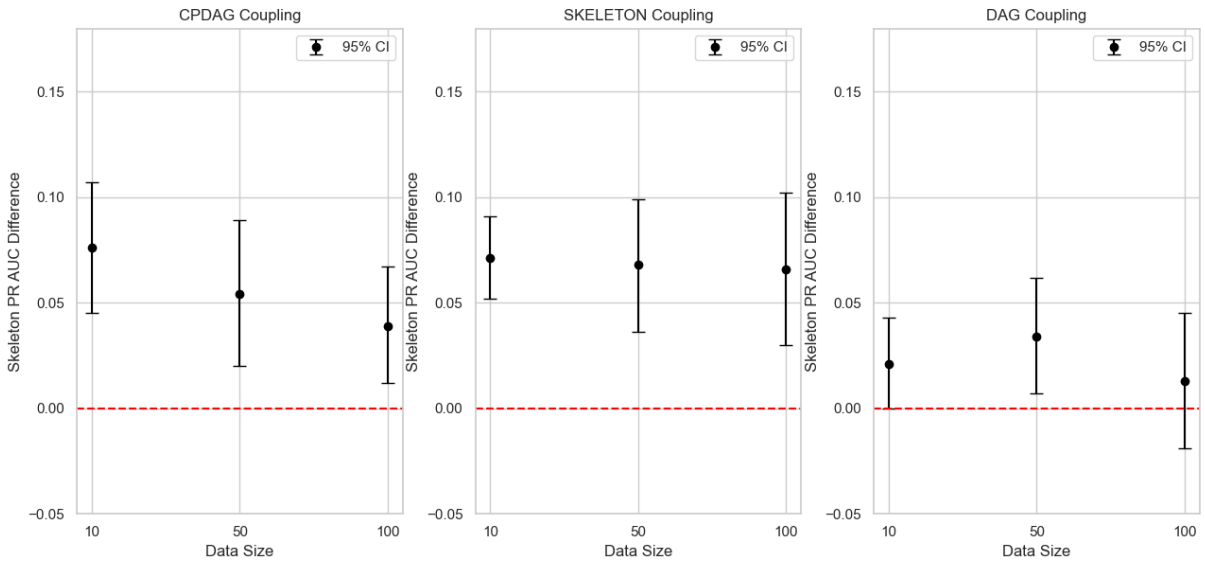


Figure 5.9: The figure shows the 95% Confidence Intervals for the network-wise differences between the coupled methods and the uncoupled method. The evaluation metric is the Raf Skeleton PR AUC.

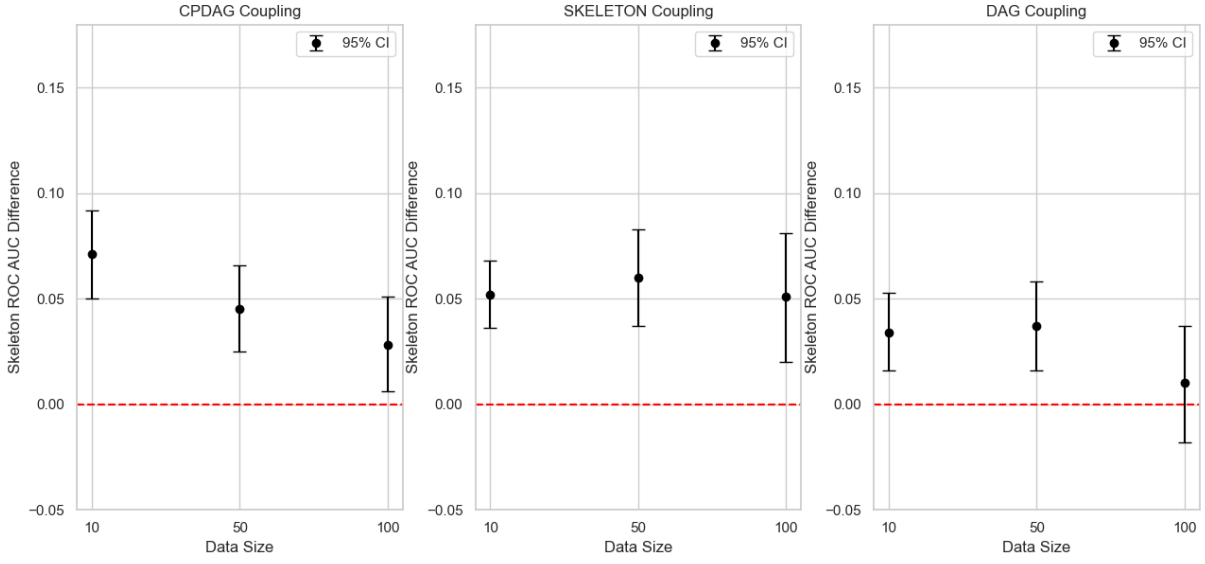


Figure 5.10: The figure shows the 95% Confidence Intervals for the network-wise differences between the coupled methods and the uncoupled method. The evaluation metric is the Raf Skeleton ROC AUC.

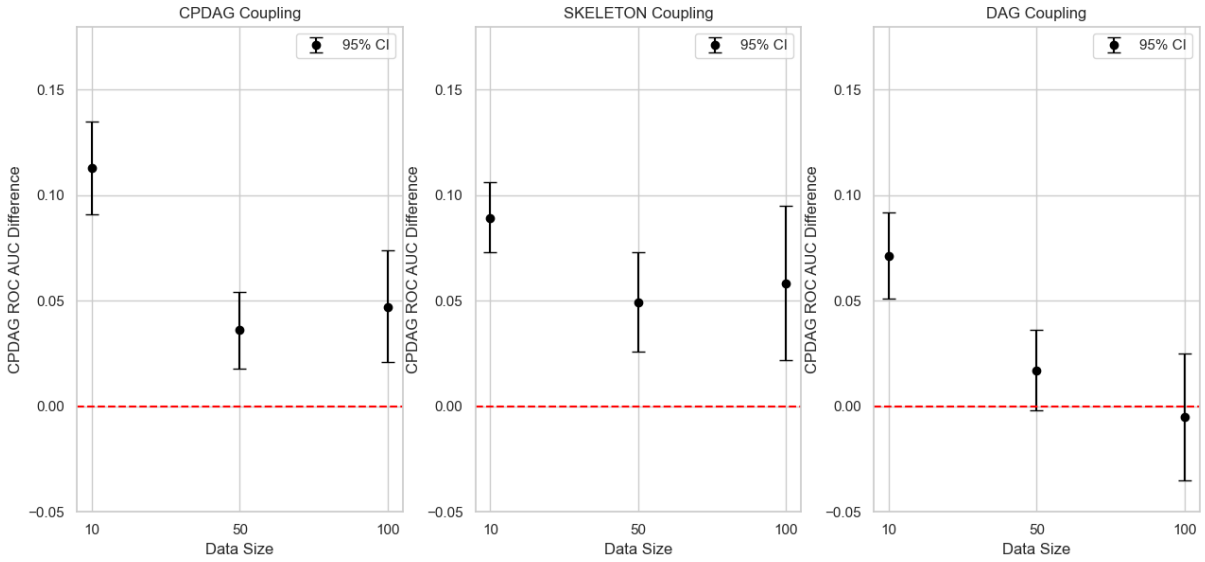


Figure 5.11: The figure shows the 95% Confidence Intervals for the network-wise differences between the coupled methods and the uncoupled method. The evaluation metric is the Raf CPDAG ROC AUC.

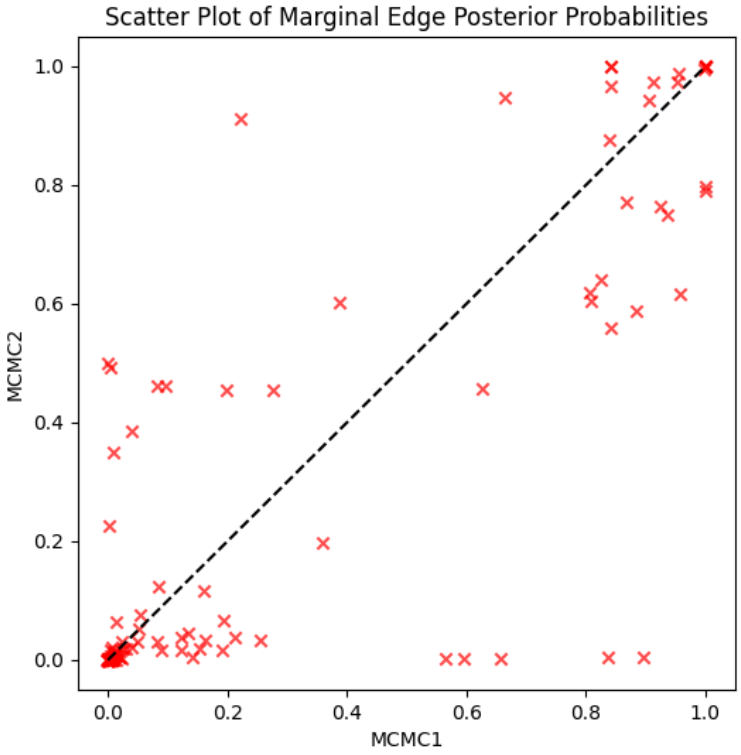


Figure 5.12: Scatter plots showing the marginal edge posterior probabilities between two different MCMC experiments present a good convergence indication. This figure shows a typical situation in our simulations when a lack of convergence of the MCMC was observed. Here, we perform two independent MCMC realizations using one of our coupled schemes and visualize the edge scores resulting from averaging one of the network samples corresponding to the dataset $\mathcal{D}_i$. Notice that the marginal posterior probabilities are often placed away from the diagonal line.

References

[1] Joaquín Abellán, Manuel Gómez-Olmedo, Serafín Moral, et al. Some Variations on the PC Algorithm. In *Probabilistic Graphical models*, pages 1–8. Citeseer, 2006.

[2] Silvia Acid and Luis M de Campos. Searching for Bayesian Network Structures in the Space of Restricted Acyclic Partially Directed Graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

[3] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network Biology: Understanding the Cell's Functional Organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[4] Dimitris Bertsimas and John Tsitsiklis. Simulated Annealing. *Statistical Science*, 8(1):10–15, 1993.

[5] Magnus Bordewich. Approximating the Number of Acyclic Orientations for a Class of Sparse Graphs. *Combinatorics, Probability and Computing*, 13(1):1–16, 2004.

[6] Thomas Brylawski and James Oxley. The Tutte Polynomial and its Applications. *Matroid applications*, 40:123–225, 1992.

[7] Baoping Cai, Lei Huang, and Min Xie. Bayesian Networks in Fault Diagnosis. *IEEE Transactions on Industrial Informatics*, 13(5):2227–2240, 2017.

[8] Andrés Cano, Manuel Gómez-Olmedo, and Serafın Moral. A Score Based Ranking of the Edges for the PC Algorithm. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models*, pages 41–48, 2008.

[9] Olivier Cappé, Arnaud Guillin, Jean-Michel Marin, and Christian P Robert. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929, 2004.

[10] David Maxwell Chickering. Learning Equivalence Classes of Bayesian-Network Structures. *The Journal of Machine Learning Research*, 2:445–498, 2002.

[11] David Maxwell Chickering. A Transformational Characterization of Equivalent Bayesian Network Structures. *arXiv preprint arXiv:1302.4938*, 2013.

[12] David P Clark and Nanette J Pazdernik. *Molecular Biology*. Elsevier, 2012.

[13] Mary Kathryn Cowles and Bradley P Carlin. Markov Chain Monte Carlo Convergence Aiagnostics: A Comparative Review. *Journal of the American statistical Association*, 91(434):883–904, 1996.

[14] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240, 2006.

[15] Martijn de Jongh and Marek J Druzdzel. A Comparison of Structural Distance Measures for Causal Bayesian Network Models. *Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science Series*, pages 443–456, 2009.

[16] Daniel A De Souza, Diego Mesquita, Samuel Kaski, and Luigi Acerbi. Parallel MCMC without Embarrassing Failures. In *International Conference on Artificial Intelligence and Statistics*, pages 1786–1804. PMLR, 2022.

[17] Byron Ellis and Wing Hung Wong. Learning Causal Bayesian Network Structures from Experimental Data. *Journal of the American Statistical Association*, 103(482):778–789, 2008.

[18] Lucia Falzon. Using Bayesian Network Analysis to Support Centre of Gravity Analysis in Military Planning. *European Journal of Pperational Research*, 170(2):629–643, 2006.

[19] Jing Yuan Fang and Bruce C Richardson. The MAPK Signalling Pathways and Colorectal Cancer. *The Lancet Oncology*, 6(5):322–327, 2005.

[20] Sacha Friedli and Yvan Velenik. *Statistical Mechanics of Lattice Systems: A Concrete Mathematical Introduction*. Cambridge University Press, 2017.

[21] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian Networks to Analyze Expression Data. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, pages 127–135, 2000.

[22] Gaudenz Alder. Draw.io - Online Diagram Software, 2024. URL `https://www.diagrams.net/`. Accessed: March 14, 2025.

[23] Dan Geiger and David Heckerman. Learning Gaussian Networks. In *Uncertainty in Artificial Intelligence*, pages 235–243. Elsevier, 1994.

[24] Alan E Gelfand. Gibbs Sampling. *Journal of the American Statistical Association*, 95(452):1300–1304, 2000.

[25] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

[26] Jayanta K Ghosh, Mohan Delampady, and Tapas Samanta. *An Introduction to Bayesian Analysis: Theory and Methods*. Springer Science & Business Media, 2007.

[27] Marco Grzegorczyk. An Introduction to Gaussian Bayesian Networks. *Systems Biology in Drug Discovery and Development: Methods and Protocols*, pages 121–147, 2010.

[28] Alexander J Hartemink, David K Gifford, Tommi S Jaakkola, and Richard A Young. Using Graphical Models and Genomic Expression Data to Statistically Validate Models of Genetic Regulatory Networks. In *Biocomputing 2001*, pages 422–433. World Scientific, 2000.

[29] W. Keith Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970.

[30] David Heckerman. Bayesian Networks for Data Mining. *Data Mining and Knowledge Discovery*, 1: 79–119, 1997.

[31] Leonard Henckel, Theo Würtzen, and Sebastian Weichwald. Adjustment Identification Distance: A gadjid for Causal Structure Learning. *arXiv preprint arXiv:2402.08616*, 2024.

[32] Linwood D Hudson, Bryan S Ware, Kathryn Blackmond Laskey, and Suzanne M Mahoney. An Application of Bayesian Networks to Antiterrorism Risk Management for Military Planners. *Digital Sand-box, Inc*, 8, 2002.

[33] Seiya Imoto, Tomoyuki Higuchi, Takao Goto, and Satoru Miyano. Error Tolerant Model for Incorporating Biological Knowledge with Expression Data in Estimating Gene Networks. *Statistical Methodology*, 3(1):1–16, 2006.

[34] Guy Karlebach and Ron Shamir. Modelling and Analysis of Gene Regulatory Networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780, 2008.

[35] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193.

[36] Andrei Nikolaevich Kolmogorov. *Foundations of the Theory of Probability*. Chelsea Pub. Co, 1933.

[37] Hugo Lavoie and Marc Therrien. Regulation of RAF Protein Kinases in ERK Signalling. *Nature Reviews Molecular Cell Biology*, 16(5):281–298, 2015.

[38] David A Levin and Yuval Peres. *Markov Chains and Mixing Times*, volume 107. American Mathematical Soc., 2017.

[39] Lei Li, Guo-Dong Zhao, Zhe Shi, Li-Li Qi, Li-Yuan Zhou, and Ze-Xian Fu. The Ras/Raf/MEK/ERK Signaling Pathway and its Role in the Occurrence and Development of HCC. *Oncology Letters*, 12(5): 3045–3050, 2016.

[40] William A Link and Mitchell J Eaton. On Thinning of Chains in MCMC. *Methods in Ecology and Evolution*, 3(1):112–115, 2012.

[41] David Madigan, Jeremy York, and Denis Allard. Bayesian Graphical Models for Discrete Data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232, 1995.

[42] Galia Maik-Rachline, Avital Hacohen-Lev-Ran, and Rony Seger. Nuclear ERK: Mechanism of Translocation, Substrates, and Role in Cancer. *International Journal of Molecular Sciences*, 20(5): 1194, 2019.

[43] Brendan D McKay, Frederique E Oggier, Gordon F Royle, NJA Sloane, Ian Murray Wanless, Herbert S Wilf, et al. Acyclic Digraphs and Eigenvalues of (0, 1)-Matrices. *Journal of Integer Sequences*, 7(2): 1–5, 2004.

[44] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[45] Radford M Neal. Annealed Importance Sampling. *Statistics and computing*, 11:125–139, 2001.

[46] Judea Peal. Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 7, 1985.

[47] Judea Pearl. From Bayesian Networks to Causal Networks. In *Mathematical Models for Handling Partial Knowledge in Artificial Intelligence*, pages 157–182. Springer, 1995.

[48] Jonas Peters and Peter Bühlmann. Structural Intervention Distance for Evaluating Causal Graphs. *Neural Computation*, 27(3):771–799, 2015.

[49] Jonas Peters, Joris M. Mooij, Dominik Janzing, and Bernhard Schölkopf. Causal Discovery with Continuous Additive Noise Models. *J. Mach. Learn. Res.*, 15(1):2009–2053, January 2014. ISSN 1532-4435.

[50] Jorge López Puga, Martin Krzywinski, and Naomi Altman. Points of Significance: Bayesian Networks. *Nature Methods*, 12(9), 2015.

[51] Charles Chapman Pugh and CC Pugh. *Real Mathematical Analysis*, volume 2011. Springer, 2002.

[52] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. Technical report, OpenAI, 2018.

[53] Josephine Rehak, Alexander Falkenstein, Frank Doehner, and Jürgen Beyerer. Metrics for the Evaluation of Learned Causal Graphs Based on Ground Truth. In *ML4CPS–Machine Learning for Cyber-Physical Systems*, 2024.

[54] Robert W Robinson. Counting Labeled Acyclic Digraphs. *New Directions in the Theory of Graphs*, pages 239–273, 1973.

[55] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science*, 308(5721): 523–529, 2005.

[56] Wanying Shi and Carlos Mena. Supply Chain Resilience Assessment with Financial Considerations: A Bayesian Network-Based Method. *IEEE Transactions on Engineering Management*, 70(6):2241–2256, 2021.

[57] Yakov G Sinai. Gibbs Measures in Ergodic Theory. *Russian Mathematical Surveys*, 27(4):21, 1972.

[58] Yanlin Song, Zhenfei Bi, Yu Liu, Furong Qin, Yuquan Wei, and Xiawei Wei. Targeting RAS–RAF–MEK–ERK Signaling Pathway in Human Cancer: Current Status in Clinical Trials. *Genes & Diseases*, 10(1):76–88, 2023.

[59] Leah F South, Marina Riabiz, Onur Teymur, and Chris J Oates. Postprocessing of MCMC. *Annual Review of Statistics and Its Application*, 9(1):529–555, 2022.

[60] Peter Spirtes, Clark Glymour, Scheines N., and Richard. *Causation, Prediction, and Search*. Mit Press: Cambridge, 1993.

[61] Richard P Stanley. Acyclic Orientations of Graphs. *Discrete Mathematics*, 5(2):171–178, 1973.

[62] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning*, 65:31–78, 2006.

[63] Aad W Van der Vaart. *Asymptotic Statistics*, volume 3. Cambridge University Press, 2000.

[64] TS Verma and Judea Pearl. *Equivalence and Synthesis of Causal Models*, page 221–236. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022.

[65] Claudia Vitolo, Marco Scutari, Mohamed Ghalaieny, Allan Tucker, and Andrew Russell. Modeling Air Pollution, Climate, and Health Data Using Bayesian Networks: A Case Study of the English Regions. *Earth and Space Science*, 5(4):76–88, 2018.

[66] Adriano V Werhli and Dirk Husmeier. Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

[67] 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*, 6(03):543–572, 2008.

[68] Adriano V Werhli, Marco Grzegorczyk, and Dirk Husmeier. Comparative Evaluation of Reverse Engineering Gene Regulatory Networks with Relevance Networks, Graphical Gaussian Models and Bayesian Networks. *Bioinformatics*, 22(20):2523–2531, 2006.

[69] Adriano Velasque Werhli. *Reconstruction of Gene Regulatory Networks from Postgenomic Data*. PhD thesis, The University of Edinburgh, 2007.

[70] Chin-Rang Yang, Bruce E Shapiro, Eric D Mjolsness, and G Wesley Hatfield. An Enzyme Mechanism Language for the Mathematical Modeling of Metabolic Pathways. *Bioinformatics*, 21(6):774–780, 2005.

[71] Chiou-Hwa Yuh, Hamid Bolouri, and Eric H Davidson. Genomic Cis-Regulatory Logic: Experimental and Computational Analysis of a Sea Urchin Gene. *Science*, 279(5358):1896–1902, 1998.

Appendix

In this Appendix, we present the results of Section 5.3. The figures show the trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(4)}$ associated with the true Raf datasets, and the most-to-the-right hyperparameter $\beta^{(5)}$ associated with the corrupted dataset.

CPDAG coupling with $|\mathcal{D}_i| = 10$

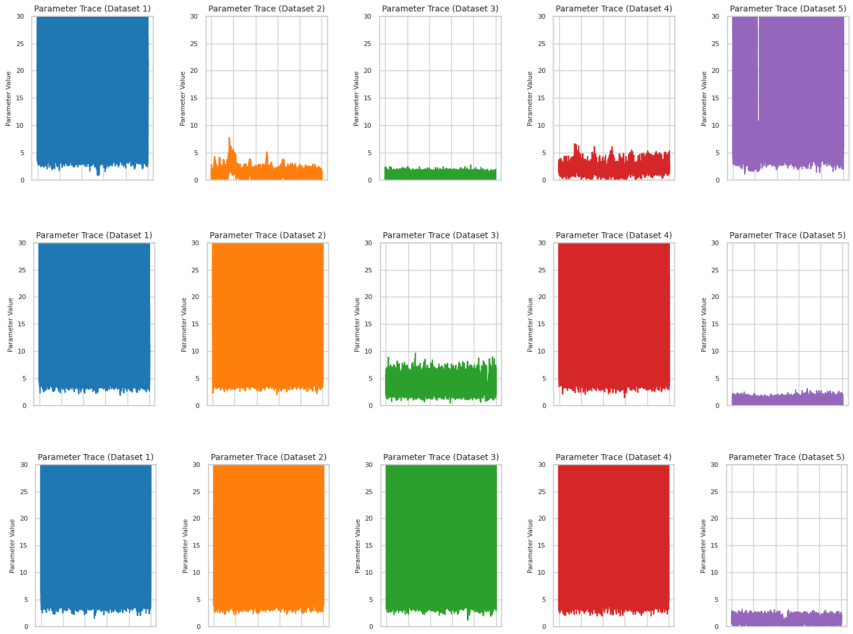


Figure .1: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of CPDAG coupling with $|\mathcal{D}_i| = 10$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.

CPDAG coupling with $|\mathcal{D}_i| = 50$

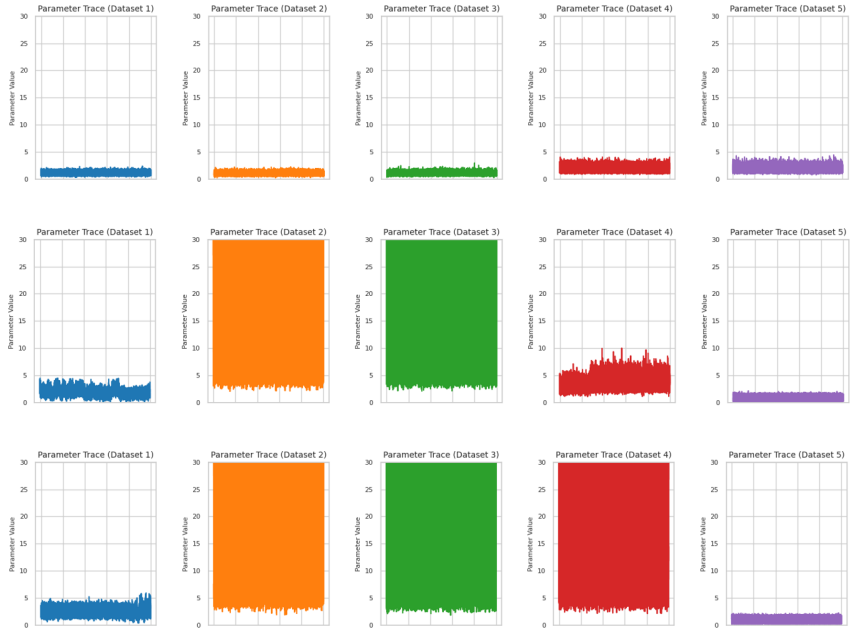


Figure .2: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of CPDAG coupling with $|\mathcal{D}_i| = 50$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.

CPDAG coupling with $|\mathcal{D}_i| = 100$

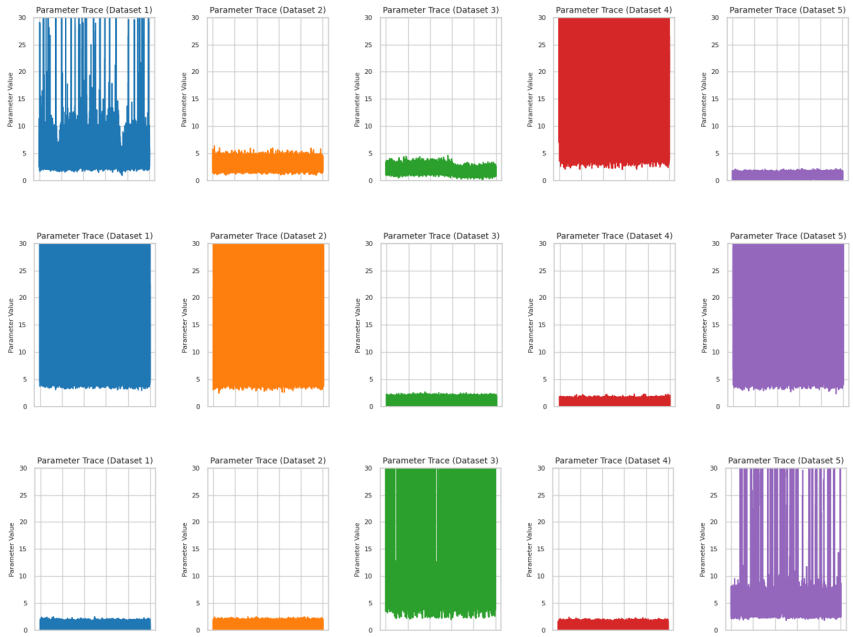


Figure .3: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of CPDAG coupling with $|\mathcal{D}_i| = 100$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.

SKELETON coupling with $|\mathcal{D}_i| = 10$

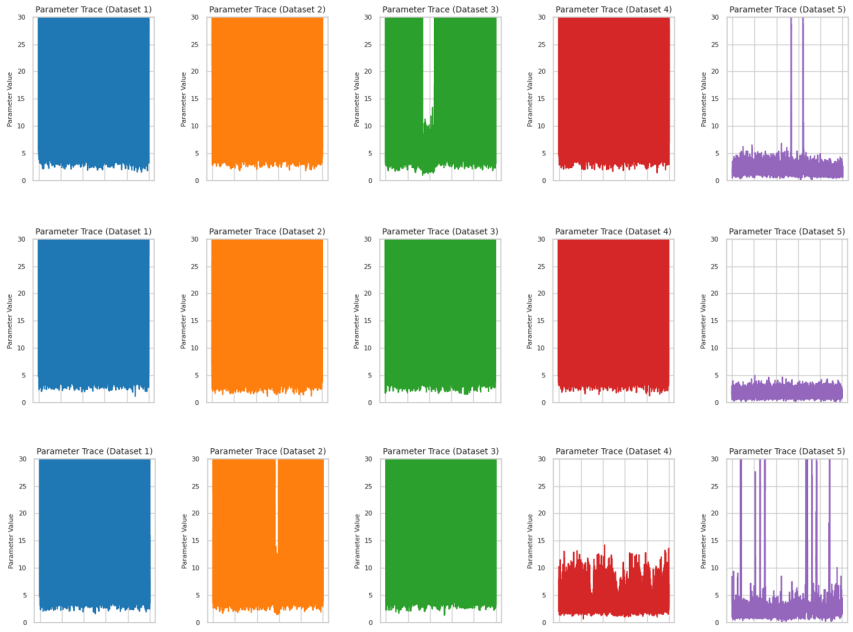


Figure .4: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of SKELETON coupling with $|\mathcal{D}_i| = 10$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.

SKELETON coupling with $|\mathcal{D}_i| = 50$

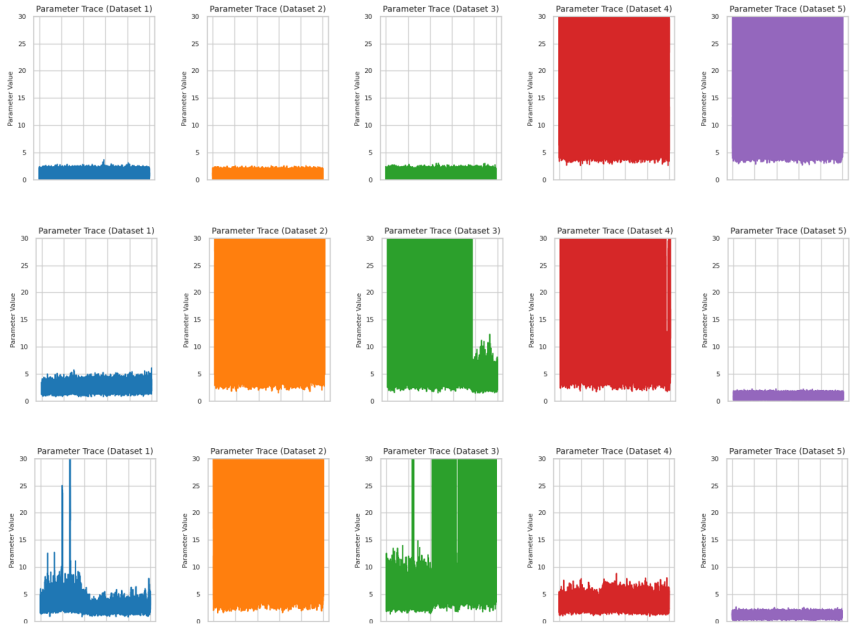


Figure .5: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of SKELETON coupling with $|\mathcal{D}_i| = 50$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.

SKELETON coupling with $|\mathcal{D}_i| = 100$

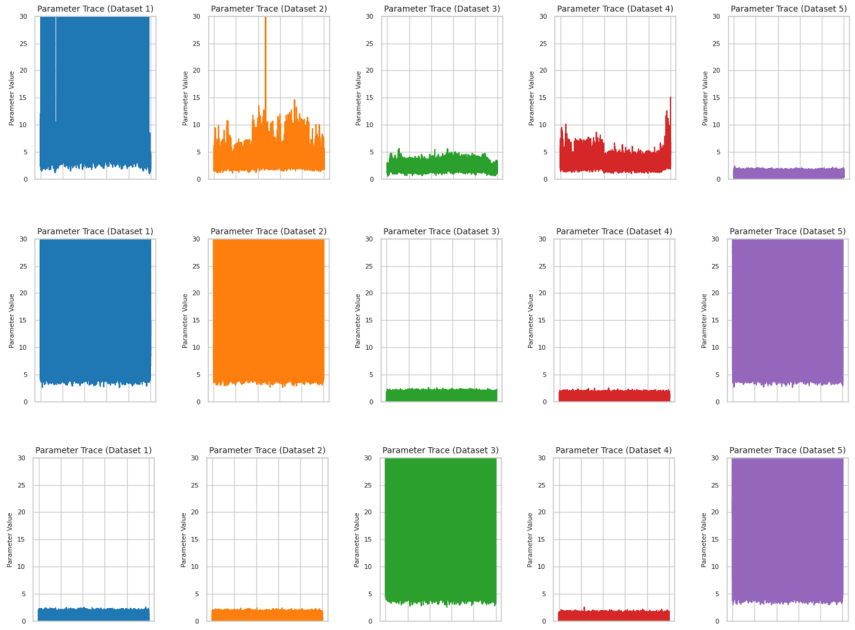


Figure .6: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of SKELETON coupling with $|\mathcal{D}_i| = 100$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.

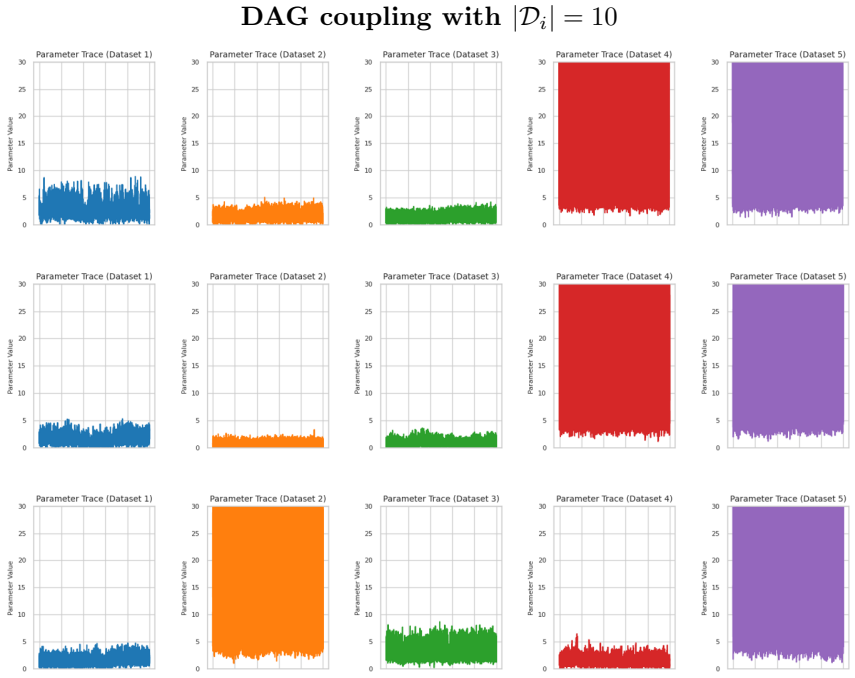


Figure .7: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of DAG coupling with $|\mathcal{D}_i| = 10$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.

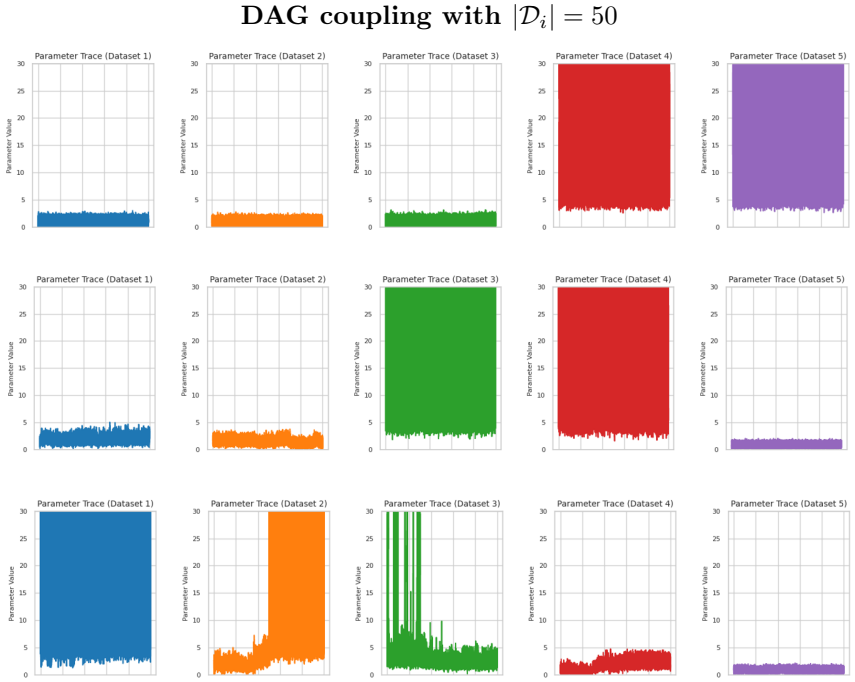


Figure .8: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of DAG coupling with $|\mathcal{D}_i| = 50$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.

DAG coupling with $|\mathcal{D}_i| = 100$

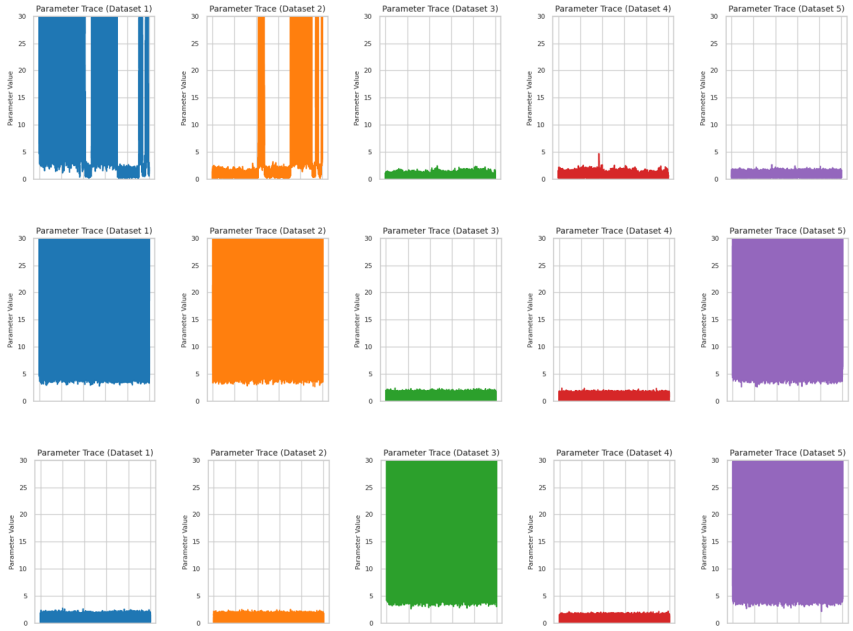


Figure .9: Trace plots of the hyperparameters $\beta^{(1)}, \ldots, \beta^{(5)}$ of DAG coupling with $|\mathcal{D}_i| = 100$. Each row corresponds to a different random seed. The last column shows the trace plots of $\beta^{(5)}$, which is the hyperparameter associated with the corrupted dataset $\mathcal{D}_5$.