Master's thesis

Master's Programme in Theoretical and Computational Methods

Enhancing Bayesian Network Discovery by Sampling Markov Equivalent DAGs

Alex Zhilkin

13th January 2025

Supervisor(s): Prof. Mikko Koivisto, Dr. Juha Harviainen

Examiner(s): Prof. Mikko Koivisto, Dr. Juha Harviainen

University of Helsinki

Faculty of Science

P. O. Box 64 (Gustaf Hällströmin katu 2)

00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Theoretical and Computational Methods

Tekijä — Författare — Author

Alex Zhilkin

Työn nimi — Arbetets titel — Title

Enhancing Bayesian Network Discovery by Sampling Markov Equivalent DAGs

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	13th January 2025	43

Tiivistelmä — Referat — Abstract

Inferring the structure of Bayesian networks from data plays a pivotal role in uncovering associations and dependencies within complex systems. Markov Chain Monte Carlo (MCMC) methods are popular for this task, as they allow computationally feasible exploration of the vast space of possible network structures by sampling from the posterior distribution. However, these methods often suffer from slow convergence, highlighting the need for more sophisticated "steps" within the Markov chain.

Building on a recently developed polynomial-time algorithm for counting and sampling Markov equivalent Directed Acyclic Graphs (DAGs), this thesis introduces the novel use of a Markov Equivalent Step (MES) within Bayesian structure discovery MCMC, leveraging the polynomial complexity of the step to improve convergence by increasing the probability of escaping local maxima and enabling a more comprehensive exploration of the posterior distribution. Experiments on small benchmark networks demonstrate that incorporating MES significantly accelerates convergence, particularly when combined with the powerful New Edge Reversal step. Furthermore, we integrate MES into the more modern Partition MCMC framework, hypothesizing that sampling an equivalent DAG can lead to distinct partitions and larger transitions within the domain of network structures, thereby enabling better exploration of the search space. Finally, testing on a larger, more challenging network shows that Partition MCMC with MES outperforms basic MCMC methods, suggesting that MES-based approaches offer a promising direction for scalable and robust Bayesian network structure learning.

ACM Computing Classification System (CCS): Computing methodologies → Artificial intelligence → Knowledge representation and reasoning → Bayesian networks
Computing methodologies → Machine learning → Machine learning approaches → Markov chain Monte Carlo (MCMC)

Avainsanat — Nyckelord — Keywords

Bayesian Network, MCMC, Markov Equivalence Class

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

https://github.com/Sums-of-Products/emci

Contents

1. Introduction

In machine learning, designing algorithms to create models from data is a central task. These models must capture the essential features of phenomena to predict future behavior or understand their structure. Advances in artificial intelligence have been driven by improvements in machine learning and the growing availability of real-world data and computational power.

Bayesian networks, which are typically constructed by domain experts [Jensen and Nielsen, 2007], can also be built using machine learning techniques. There are two main approaches to learning Bayesian networks: constraint-based and score-based. Constraint-based learning involves identifying conditional independencies through statistical tests and constructing a network representation accordingly. Score-based learning assigns scores to network structures based on their fit to the data, simplicity, and other criteria, then selects the highest-scoring structures. This thesis focuses on the latter approach, where given training data, the goal is to learn both the structure and parameters of a Bayesian network.

Structure learning is typically the most challenging aspect of the learning process, whereas parameter estimation is more straightforward once the structure is established [Koller and Friedman, 2009]. A key problem is finding the structure that maximizes the score of a Bayesian network, which is then used as a structure for inference. However, relying solely on this optimal structure can be biased, and sampling from the posterior distribution offers a more comprehensive approach.

For score-based learning, we use *Markov Chain Monte Carlo* (*MCMC*) methods [Madigan and York, 1995] to efficiently search the space of possible Bayesian network structures. Instead of exhaustively enumerating each possibility, MCMC explores this space by approximately sampling from the posterior distribution of DAGs, avoiding the focus on a single optimal structure. This approach helps in capturing a broader range of high-scoring structures, reducing bias, and providing a more comprehensive understanding of the potential models. By constructing a Markov chain where each state represents a network structure, MCMC allows us to estimate the probability distribution over structures, offering a powerful schema for structure learning of Bayesian networks.

Since its introduction, MCMC has become a widely used method for exploring the vast space of possible network structures. However, it often struggles with challenges such as getting "trapped" in sub-optimal solutions or local maxima. To address these issues, strategies such as *Order-MCMC* [Friedman and Koller, 2000] have been proposed, which improve mixing and convergence but introduce bias. Other approaches involve novel steps in the Markov chain like the New Edge Reversal [Grzegorczyk and Husmeier, 2008] or a complete restructuring based on partitions [Kuipers and Moffa, 2017]. For smaller systems, dynamic programming provides a space-efficient alternative and was the first approach to achieve less than super-exponential time complexity [Koivisto and Sood, 2004]. When the sampling distribution is modular, meaning the joint probability can be decomposed into smaller components, an exact sampler is also available, achieving a time complexity of $\tilde{O}(3^n)$ as shown at [Talvitie et al., 2020].

This thesis aims to enhance the toolkit for Bayesian network structure discovery using MCMC methods. It investigates the hypothesis, originally proposed by Jack Kuipers, a research collaborator of Mikko Koivisto (this thesis supervisor), that sampling Markov equivalent DAGs—graphs that encode the same conditional independencies and share the same skeleton and v-structures—can significantly improve convergence to the target distribution in both structural and partition MCMC. Markov equivalence classes are crucial because they capture all DAGs that encode the same conditional independencies [Heckerman and Geiger, 1995]. The practical application of these equivalence classes was previously constrained by the lack of a polynomial-time algorithm for efficient sampling [He et al., 2013]. We will explore the potential impact of a newly developed polynomial-time algorithm that addresses this challenge [Wienöbst et al., 2021].

To systematically address this hypothesis, the structure of this work progresses logically from theoretical foundations to experimental analysis. Chapter 2 introduces key concepts in Bayesian networks, followed by Chapter 3, which motivates the use of MCMC methods for structure learning. Chapter 4 discusses the recently developed polynomial-time algorithm for sampling Markov equivalent DAGs, which serves as the focal point for the experimental work conducted in this thesis.

The focus then shifts to empirical analysis in Chapters 5 and 6. Chapter 5 evaluates the effectiveness of incorporating this algorithm into structural MCMC using benchmark datasets. By analyzing trace plots and edge probability matrices, the chapter investigates the impact of sampling Markov equivalent DAGs on structural learning performance. Chapter 6 extends this analysis to Partition MCMC [Kuipers and Moffa, 2017], exploring the computational trade-offs and benefits of integrating a Markov equivalent DAG sampler into this framework. Together, these

chapters provide a comprehensive evaluation of the practical utility of this sampling technique across networks of varying complexity.

To support these empirical analyses, the experiments and methods described in this thesis were implemented using a custom tool *EMCi*, developed by the author with valuable feedback and support from members of the Sum Of Products research group at the University Of Helsinki. The full code is publicly available at https://github.com/Sums-of-Products/emci to ensure transparency and reproducibility.

2. Preliminaries

In this section, I will lay the foundation for the concepts and techniques central to this thesis. We begin by introducing the fundamental principles of graph theory, particularly focusing on Directed Acyclic Graphs and their role in Bayesian networks. Following this, I will explain the structure of Bayesian networks, including how they model conditional dependencies between random variables. A key focus will be on scoring methods for DAGs and the concept of Markov Equivalence Classes. These preliminaries provide the necessary context for understanding the MCMC-based methods discussed in subsequent chapters.

2.1 Graphs

A *graph* G is a pair (V, E), where V is a set of nodes (or vertices) and E is a set of edges connecting the nodes. In a directed graph, each edge $e \in E$ has a direction, going from one node to another. When such a graph contains no directed cycles, it is called a *Directed Acyclic Graph* (*DAG*).

In the context of a DAG, a directed edge from node u to node v indicates that u is a *parent* of v and v is a *child* of u. A *topological ordering* of nodes extends this concept by arranging them in a linear order such that for every directed edge $u \to v$, node u precedes node v in the order.

A *clique* in an undirected graph is a subset of nodes such that every two distinct nodes are adjacent, forming a complete sub-graph. For directed graphs, cliques are often identified in the moralized graph, which is constructed by replacing directed edges with undirected edges and adding edges between all parents of each node to ensure they are fully connected.

2.2 Bayesian Networks

Bayesian networks [Pearl, 1988] are probabilistic graphical models that represent a set of variables and their conditional dependencies through a DAG. In a Bayesian network,

each node corresponds to a random variable, and the directed edges depict conditional dependencies between these variables. Formally, a Bayesian network is made up of the following:

- A DAG $G = (V, E)$ where V represents random variables and E represents directed edges that indicate dependencies.
- A set of conditional probability distributions corresponding to each node, the probability of the variable's values given the values of its parents.

Let $X = \{X_1, \ldots, X_n\}$ be n random variables, which can be discrete, continuous, or a mix of both. In the discrete case, variables receive a limited set of values; in the simplest form, this could be a boolean variable with values 'yes' or 'no.' In the continuous case, variables can take on any value within a specified range, such as real numbers within an interval. This allows for a continuous range of possible values, providing a finer resolution that can capture subtle variations in data that discrete variables cannot. However, working with continuous variables can be more complex, as they often require advanced mathematical methods and integrals for probabilistic calculations.

The goal is to characterize the joint probability distribution of the variables using directed graphs. Given a set of parents Pa_i, the Markov assumption [Koller and Friedman, 2009] states that each variable X_i, conditioned on its parents, is independent of its non-descendants. Thus, we can rewrite the joint probability as:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa_i)$$

2.2.1 DAG Scoring

Scoring a graphical structure is often derived from a Bayesian framework, where the score for a DAG G is defined as its posterior probability given the data D:

$$P(G|D) \propto P(D|G)P(G)$$

where $P(G)$ represents the prior distribution over the set of possible graphical structures. The marginal likelihood $P(D|G)$ is calculated by integrating the likelihood function $P(D|G, \theta)$ over the parameter prior $P(\theta|G)$ for a particular graph G and across the parameter space θ. When the prior satisfies the conditions of parameter independence as well as structural and parameter modularity, as defined by [Heckerman and Geiger, 1995], and the data is complete, the score becomes structure-equivalent and decomposable into products [Friedman et al., 2000]:

$$P(G|D) \propto P(D|G)P(G) = \prod_i S(X_i, \mathrm{Pa}_i|D)$$

where S is a score function that depends exclusively on the node variable X_i and its parent nodes.

Decomposability is crucial for implementation as it allows evaluation at the level of individual nodes, a requirement for certain methods such as the *REV* move presented in Section 3.2. Given that the number of possible DAGs increases super exponentially with the number of nodes, an exhaustive search quickly becomes infeasible, even for a moderate number of nodes. Consequently, state-of-the-art methods often rely on approximate solutions, particularly those based on MCMC simulation techniques outlined in Section 3.1.

Numerical values for the score function $S(X_i, \mathrm{Pa}_i|D)$ can be assigned using methods such as the *Bayesian Dirichlet equivalent* (BDe) and *Bayesian Dirichlet equivalent uniform* (BDeu), both introduced by [Heckerman and Geiger, 1995], as well as the *Bayesian Information Criterion* (BIC) [Schwarz, 1978]. The BDe and BDeu scores are widely used in the context of Bayesian network structure learning, where they measure how well a given network structure fits the observed data. To calculate these scores, the concept of marginal likelihoods, which integrates over possible parameter values. The BDe score is computed by considering a Dirichlet prior over the parameters and integrating out the parameters, resulting in a closed-form expression that depends on the counts of the data and the hyper-parameters of the prior.

The BIC is another widely used scoring method. It balances the model fit with its complexity by penalizing the number of parameters in the model. This makes BIC particularly useful for model selection, as it tends to prefer simpler models that avoid overfitting, especially when the dataset is large. Unlike the BDe and BDeu scores, which rely on Bayesian priors, BIC is purely based on the likelihood of the data and the complexity of the model.

However, only BDe and BDeu adhere to the principle of *score equivalence*, which leads us to the introduction of Markov Equivalence Classes in Section 2.3. In short, the structure of the network, or more precisely the DAG, is identifiable only up to an equivalence class, and score equivalence ensures that all DAGs within the same equivalence class receive the same score.

When considering priors for our case, we will focus exclusively on modular ones, as they can be applied to individual nodes rather than being restricted to the entire graph. In a Bayesian network, assigning a prior that penalizes nodes with many parents is valuable to prevent overfitting [Heckerman and Geiger, 1995], which occurs when a model becomes overly complex and learns patterns from the training data that do not generalize well to new, unseen data. By penalizing networks with too many parent nodes — or, in other words, too many dependencies — the prior encourages simpler

Table 2.1: Commonly used modular structure priors

Prior Name	**In-degree Factor**	**Description**	**Reference**
Uniform	1	Uniform distribution over DAGs.	—
Edge	β^k	Similar to the random graph model $p^k(1-p)^{n-1-k}$.	[Heckerman and Geiger, 1995]
Fair	$\frac{1}{\binom{n-1}{k}}$	Ensures fair distribution of in-degree probabilities.	[Friedman and Koller, 2000]
Data	$\exp\left[-(1+\tau)^k \ln N\right]$	Depends on data size N, typically $\tau = 0.5$.	[Pensar et al., 2018]

models that focus on the most important relationships in the data, thus enhancing the model's generalization ability and reducing the risk of overfitting.

An extensive summary of structure priors for learning Bayesian networks is provided in [Eggeling et al., 2019]. Table 2.1, adapted from this work, compares a selection of modular priors. In this context, k represents the number of parents for a given node, n is the total number of nodes in the network, and N denotes the number of data points.

2.3 Markov Equivalence Class

A *Markov equivalence class* (*MEC*) consists of all Directed Acyclic Graphs (DAGs) that represent the same conditional independencies. Two DAGs are Markov equivalent if they share the same skeleton (an undirected version of the graph) and identical v-structures.

A v-structure, or immorality as depicted in Figure 2.1, is identified in a DAG when two non-adjacent nodes have a common child. This structure is central because it imposes a conditional dependency that would not be apparent from an undirected perspective. For example, consider a scenario where G represents a person's genetic predisposition, L represents their lifestyle choices, and H represents a health condition, such as high blood pressure. In a v-structure $G \rightarrow H \leftarrow L$, G and L are initially independent if not conditioned by anything; knowing a person's genetic predisposition tells us nothing about their lifestyle choices, and vice versa. However, given the presence of high blood pressure (conditioning on H), G and L become conditionally dependent. If we know that someone with high blood pressure has a genetic predisposition (G),

it reduces the likelihood that their lifestyle (L) is the primary cause, and vice versa. This is significant because, in Markov networks, dependencies usually require a direct connection between nodes. However, in DAGs, the v-structure allows for conditional dependencies through indirect connections, specifically when conditioning on the common child. This feature is crucial in Bayesian networks, as it defines how information flows through the network and affects dependency relationships, which impacts both inference and learning processes.

A DAG with a big equivalence class is particularly significant because it may represent a region in the search space with high posterior probability. Sampling from these equivalent DAGs allows an MCMC algorithm to more thoroughly explore this region, potentially leading to more accurate Bayesian network structures.

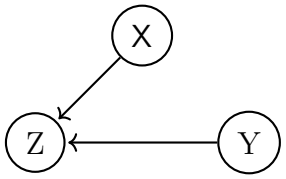

Figure 2.1: $X \to Z$ and $Y \to Z$ form a v-structure.

Determining the size of each MEC and sampling from it is highly beneficial for Bayesian structure discovery. Focusing on MECs rather than individual DAGs enhances model learning and facilitates causal analysis [Andersson et al., 1997]. This focus has spurred extensive research into the properties and applications of Markov equivalence classes of DAGs [He et al., 2015, Talvitie and Koivisto, 2019].

3. Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods form a class of algorithms that approximate probability distributions by generating samples from them. It does this by constructing a Markov chain, a sequence of random states where each state depends solely on the previous one. Over time, this chain converges to a stationary distribution that matches the target distribution. By sampling from this chain, we can estimate properties of the target distribution without needing to compute it directly, which is especially valuable for high-dimensional or complex distributions.

In Bayesian network structure learning, MCMC plays a key role in sampling from the posterior distribution of possible network structures. Since the number of DAGs grows super-exponentially with the number of variables, exhaustive search is impractical. Moreover, determining a Bayesian network where each node has at most K parents and a posterior probability exceeding a given constant is proven to be NP-complete [Chickering, 1996]. Instead, we define a Markov chain over the posterior probability distribution $P(G|D)$ of the network G given the data D, enabling us to explore the space of likely structures through sampling rather than enumeration [Madigan and York, 1995].

The following section describes how MCMC is applied in Bayesian structure discovery by defining a neighborhood of DAGs for each structure. Basic moves, such as adding, removing, or reversing edges, are used to construct this neighborhood, creating a Markov chain that allows for exploration of possible structures. Additionally, advanced moves such as the *New Edge Reversal* (*REV*) [Grzegorczyk and Husmeier, 2008] move expand the chain's ability to escape local maxima by introducing new v-structures, facilitating efficient exploration of the structure space.

3.1 Bayesian Network Structure Learning

As discussed previously, the number of Bayesian network structures is super-exponential in the number of variables in the domain. Markov Chain Monte Carlo is a well-established method for approximating samples from the posterior distribution. We define a Markov chain over the posterior probability distribution $P(G|D)$ of

the network G provided the data D.

Given a DAG G_i, to sample a proposed DAG G' using basic moves, we must first identify the neighborhood $N(G_i)$, which consists of all valid DAGs that can be reached by adding, removing, or reversing an edge. This process is referred to in this thesis as the *3-step* chain. The size of this neighborhood is denoted by $\#N(G_i)$. We then sample uniformly from this set, with the probability of proposing a specific DAG G' from G_i by

$$q(G'|G) = \begin{cases} \frac{1}{\#N(G_i)} & \text{if } G' \in N(G_i) \\ 0 & \text{otherwise} \end{cases}$$

First introduced in the seminal paper [Madigan and York, 1995], the acceptance probability of proposed DAG is thus:

$$\rho = \min\left\{1, \frac{q(G_i|G')P(G'|D)}{q(G'|G_i)P(G_i|D)}\right\} = \min\left\{1, \frac{\#N(G_i)P(G'|D)}{\#N(G')P(G_i|D)}\right\}$$

Such that the next DAG in the chain is the proposed G' with probability ρ, or G_i otherwise. This procedure follows the Metropolis-Hastings algorithm [Hastings, 1970], a Markov Chain Monte Carlo (MCMC) method that allows for sampling from complex probability distributions by constructing a Markov chain whose stationary distribution is the target posterior distribution.

To avoid calculating the neighborhood at each step of the chain we can use a novel "rejection sampling" approach (Juha Harviainen, University Of Helsinki, July 2023) outlined in the pseudo-code of Algorithm 1.

Algorithm 1 Rejection Sampling MCMC

Require: A DAG $G = (V, E)$

Ensure: A new DAG G'

1: **Sample** nodes v and $u \in G$ with $v \neq u$ uniformly.
2: **if** $(v, u) \in E$ **then**
3: Remove the edge (v, u).
4: **else if** $(v, u) \in G$ **and** reversing it to (u, v) does not create a cycle **then**
5: Apply the reverse: $(v, u) \to (u, v)$.
6: **else if** $(v, u) \notin G$ **and** $(u, v) \notin G$ **and** adding (v, u) would not create a cycle **then**
7: Add a new edge (v, u).
8: **else**
9: Set the proposal equal to the current DAG G.
10: **end if**
11: **Accept** the candidate G' with probability

$$\rho = \min \left\{ 1, \frac{P(G')P(G' \mid D)}{P(G)P(G \mid D)} \right\}$$

12: **Otherwise**, remain with G.

Theorem 1. *The proposed Algorithm 1 constructs a Markov chain that is ergodic and converges to a stationary distribution. Consequently, it forms a valid Markov chain.*

Proof. First, the chain is irreducible because any DAG can be reached from any other DAG by a finite sequence of edge additions, removals, or reversals. Secondly, the chain is aperiodic because there is always a non-zero probability of staying in the same state by making no change, ensuring that the chain does not follow a fixed cyclic pattern.

Then, to show the Markov chain converges to a stationary distribution, we begin by assuming it has a stationary distribution $p(G)$, where $p(G)$ represents the posterior probability of the DAG G. This probability is proportional to its score, $p(G) \propto \text{score}(G)$, which combines prior information and the likelihood of the data given G. It is known that the size of the neighborhood $N(G)$ is bounded by $n(n-1)$.

Let the random variable G_0 denote the current DAG and G_1 denote the DAG after the iteration. There are three possible ways that the DAG could be G after an iteration:

1. It transitions from one of its neighboring graphs G' (first sum).
2. It remains unchanged after proposing a different DAG (second sum).
3. The proposed DAG is the same as the current one G (last term).

Therefore, the probability of $G_1 = G$ can be expressed as:

$$\Pr(G_1 = G) = \left(\sum_{G' \in N(G)} \frac{1}{n(n-1)} \Pr(G_0 = G') \max\left\{1, \frac{p(G)}{p(G')}\right\} \right) + \Pr(G_0 = G) \left(\sum_{G' \in N(G)} \frac{1}{n(n-1)} \left(1 - \max\left\{1, \frac{p(G')}{p(G)}\right\}\right) \right) + \Pr(G_0 = G) \left(1 - \frac{|\#N(G)|}{n(n-1)}\right).$$

Simplified further as:

$$= p(G) \left(\sum_{G' \in \#N(G)} \frac{1}{n(n-1)} \max\{p(G'), p(G)\} \right) - p(G) \left(\sum_{G' \in \#N(G)} \frac{1}{n(n-1)} \max\{p(G'), p(G)\} \right) + p(G) \left(1 - \frac{|\#N(G)|}{n(n-1)}\right)$$

Finally, this simplifies to $\Pr(G_1 = G) = p(G)$ confirming that the stationary distribution is indeed preserved. □

3.2 New Edge Reversal

As we know from the Metropolis-Hastings algorithm, a good proposal distribution can help with convergence. Therefore, also in graph space we can utilize more sophisticated steps than adding, removing or reversing an edge.

A rather popular and useful one is the *New Edge Reversal* (*REV*) move [Grzegorczyk and Husmeier, 2008], it refines the structure MCMC sampler by choosing an edge, reversing its direction, and updating the parent-child relationships for the affected nodes. It enhances the MCMC's ability to explore the space of DAGs more efficiently. The REV move is considered as a powerful tool to escape local maxima. This is explained by the fact that when sampling a new parent set for each involved node, it has a high probability of introducing new v-structures leading to new equivalence classes.

On the other hand, REV is computationally intensive compared to basic steps, which can be performed without recalculating the neighborhoods at each step. It includes sampling a new parent set from all eligible ones (not creating a cycle, and one should include the other to reverse the edge). The basic implementation assigns a probability distribution based on scores for all eligible parent sets and samples accordingly; while effective, it is computationally expensive.

This method is implemented and tested in conjunction with the sampling of a Markov equivalent DAG. Its established usefulness as an enhancement to the basic 3-step process is further evaluated through experiments discussed in later sections.

4. Polynomial Time Sampling of Markov Equivalent DAGs

In the context of MCMC, incorporating a Markov Equivalent DAG sampling step (hereafter referred to as MES) can facilitate a chain's ability to "escape" local maxima. However, this approach has remained largely unexplored due to the lack of efficient algorithms for its implementation, with sampling over such equivalence classes being feasible only for graphs with fewer than approximately 20 nodes [He et al., 2013]. Sampling a new DAG within the same equivalence class provides a structure with the same score and encoded independencies but potentially positioned at a significantly different location in the space of possible DAGs. This can enable substantial "jumps" to regions where the current point is not a local maximum, allowing the chain to move forward and reach DAGs with higher scores within the relevant DAG space associated with the Bayesian network. This idea forms the foundation of the hypothesis presented in this thesis.

4.1 Completed Partially Directed Acyclic Graph

A *Completed Partially Directed Acyclic Graph* [Chickering, 2002], also referred to as an *Essential Graph* by [Andersson et al., 1997], can be used to represent a *MEC* (*Markov Equivalence Class*). Each undirected edge can be oriented in whichever way, while still keeping all the v-structures and causal independencies of a DAG intact.

The undirected components of CPDAGs are *UCCGs* (Undirected and Connected Chordal Graphs). A *Chordal Graph* is an undirected graph in which every cycle of four or more nodes has a chord (an edge connecting two non-adjacent nodes in the cycle). In other words, a graph is chordal if all its cycles of length four or greater can be broken into smaller cycles with an edge. This ensures that there are no induced cycles in the graph of length greater than three.

In Bayesian structure discovery, MCMC methods have been also explored in the space of essential graphs or CPDAGs to improve computational efficiency. [He et al., 2013] propose a reversible and irreducible Markov chain that operates di-

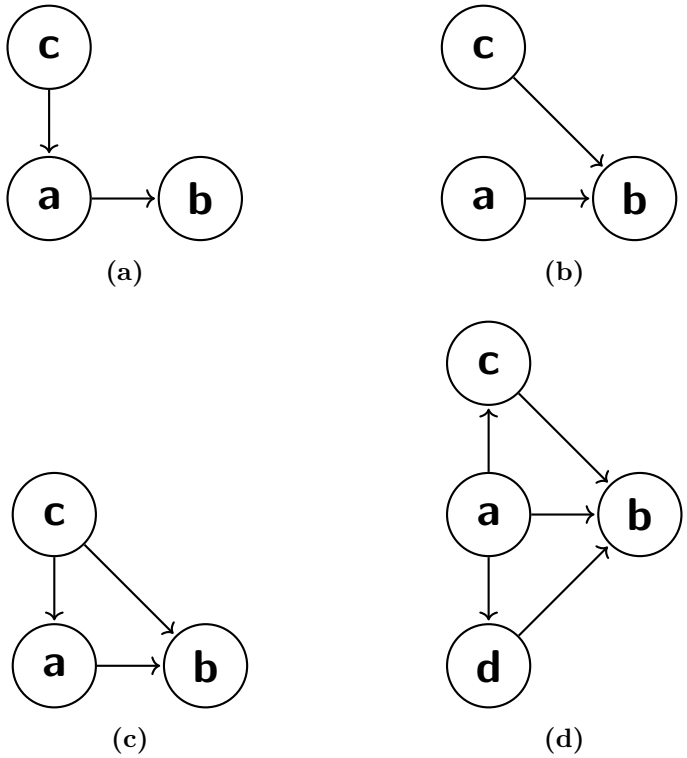

Figure 4.1: The graphs (a)-(d) depict all different cases where we have a protected edge a $\rightarrow$ b

rectly on the Markov equivalence classes of DAGs. This approach leverages the CPDAG representation to sample efficiently from the posterior distribution over equivalence classes, overcoming some of the challenges posed by the vast search space in Bayesian network learning.

4.1.1 Derivation

To compute a graph's CPDAG, we must first identify all edges that are not strongly protected. A strongly protected edge (a, b) adheres to one of the patterns shown in Figure 4.1 [Andersson et al., 1997], meaning that the direction of the edge cannot be altered without either breaking an existing v-structure or creating a new one. Edges that are not strongly protected then converted into undirected ones to form the undirected sub-graphs in a CPDAG.

4.2 Counting AMOs in Polynomial Time

The primary goal of the algorithm is to receive a CPDAG and orient its undirected edges in a way that does not create new v-structures. It is determined that for each undirected and connected chordal sub-graph (UCCG) resulting from the undirected

parts of a CPDAG we can count the Acyclic Moral Orientations (AMOs) separately and their product is equal the AMOs count of the original DAG G [Andersson et al., 1997]:

$$\#\mathrm{AMO}(G) = \prod_{H \in G} \#\mathrm{AMO}(H)$$

where H is a UCCG resulting from the undirected parts of the CPDAG of G. Therefore, the problem of counting the AMOs reduces to counting the AMOs of each UCCG.

To achieve this, as outlined in [Wienöbst et al., 2021], we need to define several steps: First, the set $C_G(K)$, which represents the connected components of the graph G constrained by the clique K and processed using a Lexicographic BFS (Breadth-First Search). This process is detailed in Algorithm 1, Section 3 of [Wienöbst et al., 2021]. Notably, every Lexicographic BFS ordering of the nodes in a UCCG corresponds to an AMO, and each such ordering begins with a maximal clique.

Second, Let S be a set and $\mathcal{R} = \{X_1, \ldots, X_\ell\}$ be a collection of subsets of S with $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_\ell$. Then we define $\phi(S, \mathcal{R})$ as the count of all S permutations that do not have a set $S' \in R$ as a prefix. This can be defined as the following recursive formula:

$$\phi(S, \mathcal{R}) = |S|! - \sum_{i=1}^{\ell} |S \setminus X_i|! \cdot \phi(X_i, \{X_1, \ldots, X_{i-1}\}). \tag{4.1}$$

Third, *Forbidden Prefixes* (FP) are nodes that we exclude from the final count when encountering a clique, as they are shared across other cliques on the path from the root and their possible orientation was accounted for. While traversing the rooted clique tree $\mathcal{T}$ with root $\mathcal{R}$ where the nodes c_i represent maximal cliques in the original graph, we define the following forbidden prefixes for a given topological order of nodes: Let P represent the path from the root R to the node v. For each clique $c_i \in P$, define the set $V(c_i)$ as the nodes in clique c_i and the set of forbidden prefixes is then given by:

$$\mathrm{FP}(v, \mathcal{T}) = V(c_i) \cap V(c_{i+1}) \subseteq V(v). \tag{4.2}$$

These steps culminate in Algorithm 2.

Algorithm 2 Count *AMO* (G)

Input: UCCG $G = (V, E)$

Output: $\#AMO(G)$

$T = (T, r, \iota) \leftarrow$ a rooted clique tree of G
$sum \leftarrow 0$
$Q \leftarrow$ queue with single element r
while Q is not empty **do**
  $v \leftarrow \text{pop}(Q)$
  push(Q, children(v))
  $prod \leftarrow 1$
  for all $H \in C_G(\iota(v))$ **do**
    $prod \leftarrow prod \cdot \texttt{count}(H)$
  end for
  $sum \leftarrow sum + \phi(\iota(v), \text{FP}(v, T)) \cdot prod$
end while
return *sum*

4.3 Sampling

Sampling is similar to counting. By utilizing the set $C_G(K)$, forbidden prefixes (Equation 4.2) and ϕ (Equation 4.1) It is possible to uniformly sample a topological order representing a Markov equivalent DAG.

Starting from the root of a clique tree, we sample a clique c, with $V(c)$ denoting the set of nodes in the graph included in it, with probability proportional to:

$$\phi(V(c), \text{FP}(c, T)) \times \prod_{H \in \mathcal{C}_G(V(c))} \#AMO(H) \tag{4.3}$$

Then, from $V(c)$, we uniformly draw a permutation without c's forbidden prefixes. This is followed by recursion over the components of $\mathcal{C}_G(V(v))$, appending each permutation to the final order. The result is a uniformly sampled topological ordering representation of a Markov equivalent DAG. From this, all that remains is to orient the undirected edges of the UCCG.

5. Structure MCMC

To assess the effectiveness of the previously presented algorithm for sampling Markov equivalent DAGs, we generate MCMC chains from benchmark dataset scores. These experiments incorporate various method combinations, including a step that samples a Markov equivalent DAG with a fixed probability. The resulting DAG chains provide a basis for comparing the performance and impact of these methods.

5.1 Benchmark Data

In this study, we utilize a Python implementation of GOBNILP [Cussens, 2020] to compute and generate score files based on benchmark data. The data points, randomly sampled using the pgmpy library [Ankan and Textor, 2023], are derived from benchmark Bayesian networks, all sourced from a repository maintained by the Hebrew University of Jerusalem [Elidan, 2001]. These sampled data points are then fed into GOBNILP, which generates score files quantifying the scores for all possible parent sets of each node.

Using these score files, the data were analyzed with the previously mentioned EMCi tool. This tool includes implementations for MCMC over Bayesian networks, incorporating additional techniques such as New Edge Reversal (REV) and Markov Equivalent Step (MES).

The selected networks represent a diverse range of variable counts. Networks with fewer than 20 variables are relatively straightforward for the basic *3-step* chain to handle, enabling it to efficiently locate the highest-scoring regions. As a result, these simpler networks will not be included in the next section utilizing trace plots but will appear in the subsequent section comparing edge probabilities. Throughout this chapter, the following networks [Elidan, 2001] will serve as the source for datasets:

- *Asia* - 8 variables
- *Sachs* - 11 variables
- *Child* - 20 variables
- *Insurance* - 27 variables

- *Alarm* - 37 variables
- *Hailfinder* - 56 variables

In addition to an empirical dataset, we use *Zoo* [Forsyth, 1990], which contains 17 variables.

5.2 Plots

The simplest way to compare different chains is by using *trace plots*. These plots show the scores of sampled DAGs over the steps of the chain. While they don't give exact results, they can help us understand where the chain has been compared to others.

We aim for our chains to achieve the highest possible score at some point during their run time. As mentioned earlier, this is particularly crucial for networks where a significant portion of the probability mass lies on moderately good DAGs, or in medium to large networks with over 20 variables. Chains that attempt to uncover the structure of these networks often struggle to find the path to the highest-scoring DAG. For networks with 20 or fewer variables, exact optimal results can be obtained more easily, for example, using augmented dyadic decision trees [Talvitie et al., 2018] or with optimal path extension [Karan and Zola, 2016]. These can serve as benchmarks for the sampled chains, which will be discussed in a later section.

Therefore, we analyze the following datasets derived from benchmark Bayesian networks: Child, Insurance and Alarm. Beginning by comparing a *basic 3-move* (adding, removing or reversing) chain to a chain with an additional MES or REV move both invoked with a probability of 1/15. Datasets of sizes 100, 1,000 and 10,000 are sampled to represent various scenarios, with 5 chains run for each dataset size and chain variation.

To offset the high computational cost of a REV step, we perform 10 steps of other moves followed by thinning. For instance, in the case of Alarm (Figure 5.3), the dotted REV chain runs for 100,000 iterations, whereas both Basic and Basic with MES chains run for 1 million iterations, with thinning applied at intervals of 10.

The naming convention for the plots follows the format: 'network-name'-'sample-size', where 'network-name' refers to the specific network being analyzed, and 'sample-size' indicates the number of samples. For example, plots for the "insurance" network with 100 and 1000 samples would be named 'insurance-100' and 'insurance-1000', while plots for the "child" network would be labeled as 'child-100' and 'child-1000'. Each plotted line represents an independent run of the chain, starting from an empty graph. In some cases, distinguishing between chains of the same type is difficult, as they oscillate within the same region and are represented by the same color.

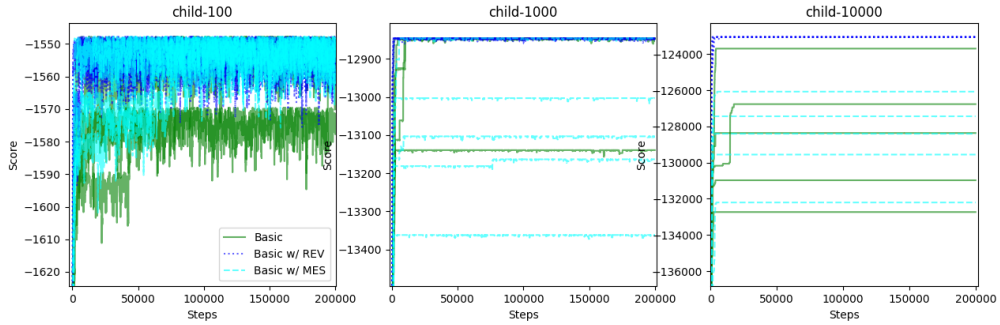

Figure 5.1: *Child* 20 variables trace plot

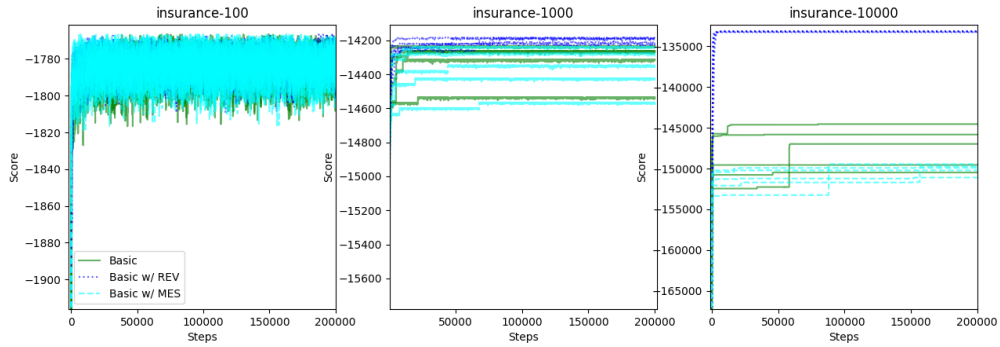

Figure 5.2: *Insurance* 27 variables trace plot

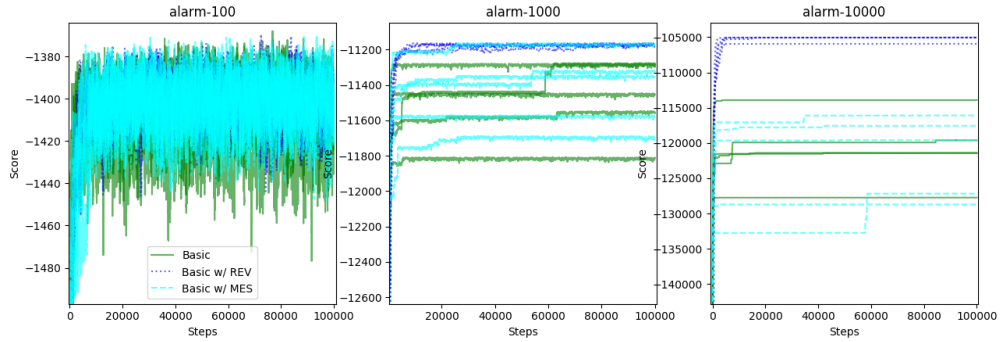

Figure 5.3: *Alarm* 37 variables trace plot

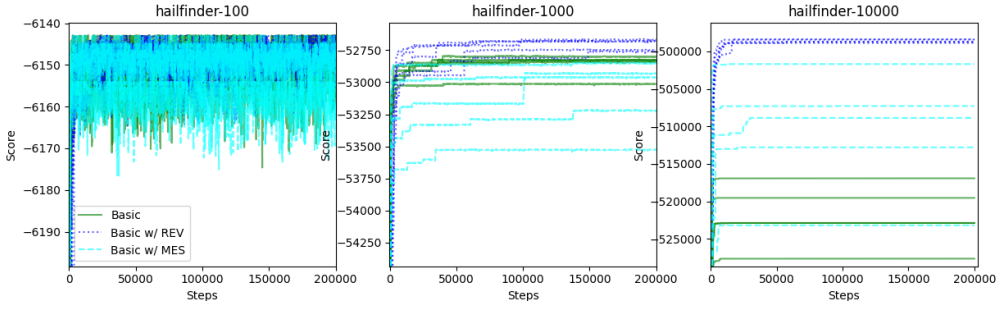

Figure 5.4: *Hailfinder* 56 variables trace plot

Analysis

In Figures 5.1, 5.2, 5.3 and 5.4 where the y-axis specifies DAGs score and the x-axis is index in the chain, there is no noticeable advantage for chains that include MES. In some plots, we could argue that the MES chain reaches higher score regions; however, the evidence remains inconclusive. In contrast, the rest show a clear advantage for the REV step, which consistently reaches higher regions with most chains, unlike the Basic.

For *hailfinder-1000* in Figure 5.4, we observe that even with the REV move, it fails to converge along the same paths. This presents an opportunity to leverage MES for potentially better results. One possible approach would be to explore combining REV and MES in a single run. We can recalculate using 5 Markov chains for each case: Basic, REV and MES with REV this time with a length of 4 million steps for Basic and ten times less for the others. Figure 5.5 shows no evidence of improvement when incorporating MES, as the REV with MES line does not show any noticeable performance advantage over REV.

It is challenging to determine why sampling a Markov equivalent DAG fails to aid in escaping a local maximum in this scenario. One plausible hypothesis, drawn from observing the DAGs in the chain, is that these DAGs tend to possess a very low number of Acyclic Moral Orientations (AMOs), often around 4 or 8. Consequently, sampling an equivalent DAG does not introduce sufficient variation, instead resulting in a repetitive cycle among the same four DAGs. Since these DAGs themselves might represent local maxima, the process is unable to facilitate progress toward better solutions. Furthermore, it is possible that these equivalent DAGs are clustered within the same 'region' of the DAG space, while the desired, better solution lies significantly farther away, rendering MES unhelpful in reaching it.

Probability of MES

By default in this section, MES is performed with a probability of 1/15, inspired by the REV step probability defined in the original paper [Grzegorczyk and Husmeier, 2008]. This choice was made as it strikes an optimal balance between computational cost and chain quality.

Unlike REV, which can with a probability approaching 1 produce a viable chain, MES lacks this capability. It does not have the potential to move towards higher score regions, and if its probability is set too high, the chain may become 'trapped', repeatedly cycling through the same N equivalent DAGs. Experimentation with various probabilities were done when trying to find a combination that will improve trace plots significantly without success. Therefore a probability of 1/15 was set.

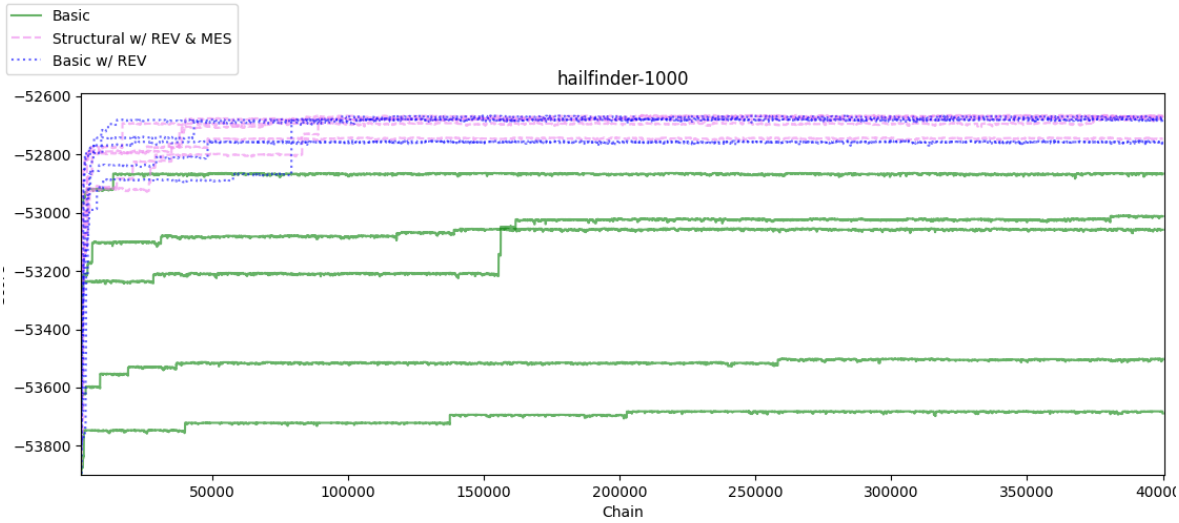

Figure 5.5: *hailfinder-1000* with *REV & MES*

As it is hard to assess how good a chain is, it is difficult also to optimise the probability of MES. This will be iterated in the next section of this thesis which deals with comparisons to exact benchmark results of smaller networks.

5.3 Edge Probabilities

Although trace plots offered valuable insights, they did not provide exact results. To address this, we will analyze datasets with a relatively small number of variables, enabling a comparison of different methods and steps in the MCMC process, where achieving exact results is more feasible. For this analysis, we chose to calculate the *edge probability matrix*, an $n \times n$ matrix that represents the probabilities of directed edges between pairs of nodes in a DAG. Each entry P_{uv} in P represents the probability of a directed edge from node u to node v in the chain, that is:

$$P_{uv} = \mathbb{P}((u, v) \in E)$$

Where $\mathbb{P}((u, v) \in E)$ is the probability that the directed edge from u to v exists in the graph. The edge probability matrix P has two key properties: $P_{uv} \in [0, 1]$ for all $u, v \in V$, as it represents a probability, and $P_{uu} = 0$ for all $u \in V$, since self-loops are not allowed in a DAG.

It provides a probabilistic description of the connectivity between nodes in a DAG. Each entry indicates the proportion of sampled DAGs in which the corresponding directed edge exists, reflecting its likelihood within the Markov chain.

As a source of truth, we first generate a benchmark matrix using the APS framework [Pensar et al., 2020], which enables the exact computation of parent set and an-

cestor relation posterior probabilities via dynamic programming. We then derive an edge probability matrix from the Markov chain of DAGs and compare the two matrices to evaluate the quality of the MCMC-derived results.

Utilizing The APS Framework

In the APS framework, for each variable and its corresponding parent set, we compute the total weight of all DAGs where a node is associated with a specific parent set. The APS method ensures exact computation of these weights by leveraging dynamic programming, allowing for the precise derivation of edge probabilities.

To construct the edge probability matrix, further processing is required. For each variable pair $(u, v) \in E$, the weights of all parent sets that include u as a parent of v are aggregated. This aggregate is then normalized over all parent sets $\mathcal{S}$ of v to produce the exact edge probabilities matrix P, defined as follows:

$$P_{uv} = \frac{\sum_{\mathcal{S} \ni u} w(\mathcal{S}, v)}{\sum_{\mathcal{S}' \subseteq V \setminus \{v\}} w(\mathcal{S}', v)}, \tag{5.1}$$

where $w(\mathcal{S}, v)$ represents the weight of its parent set for node v, and V is the set of all variables in the network. This exact computation of edge probabilities serves as the benchmark for comparing the matrix derived from the Markov chain of DAGs.

Generating From a Markov Chain

With the matrix generated by the APS framework serving as a benchmark, we proceed to derive a matrix from the Markov chain of DAGs using the procedure outlined below: Let DAG $G_j = (V, E_j)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of n nodes, and E_j is the set of directed edges in the specific DAG G_j sampled at step j of the Markov chain.

Each entry P_{uv} in P represents the probability of a directed edge from node u to node v across all sampled DAGs G_j in the Markov chain. Specifically, it is computed as:

$$P_{uv} = \frac{\text{Number of sampled DAGs where } (u, v) \in E_j}{\text{Total number of sampled DAGs}}, \tag{5.2}$$

where the numerator counts the DAGs in the Markov chain in which the edge (u, v) exists.

5.3.1 MES Probability Optimization

Determining the optimal general-purpose probability for executing a MES can be a complex challenge. Different networks are likely to benefit from different probabilities. Early in the chain, there may be little reason to 'waste' a step sampling an equivalent DAG, as other steps may already be driving the network towards higher-scoring DAGs.

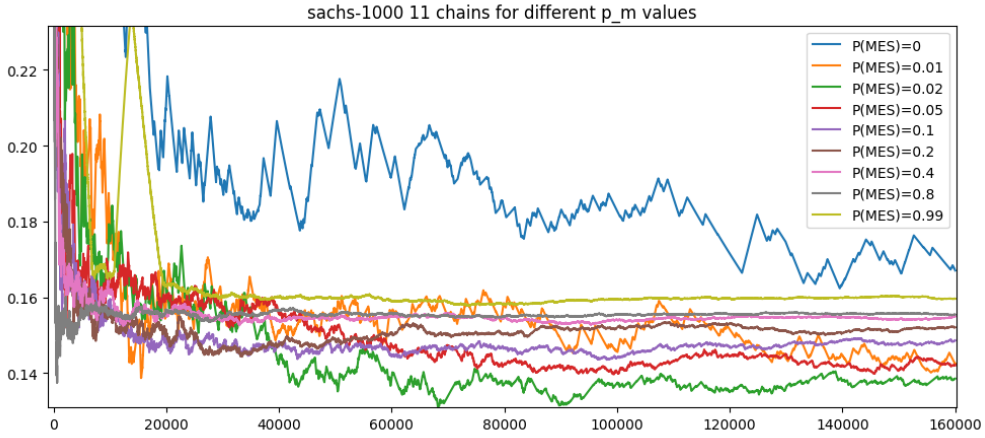

Figure 5.6: *Sachs* 11 variables error

To address this, we aim to use the exact edge probability matrix to identify a MES probability yielding the lowest maximum error.

Utilizing the Child, Sachs and Zoo datasets, we generated 11 chains for each probability value, incorporating both REV and MES, and calculated the median of the maximum errors.

As illustrated in Figure 5.6, extreme probabilities $P(MES) \in \{0, 0.99\}$ yield the poorest performance, corresponding to the highest error plots. This result aligns with expectations since very low probabilities near 0 limit the algorithm's ability to explore the search space, while very high probabilities near 0.99 overly focus on MES moves, neglecting potential refinements from other steps such as adding, removing, reversing or REV steps.

Upon further analysis, we observe that probabilities in the range of $[0.01, 0.05]$ tend to provide the best performance, with minor variations across different datasets, as seen in Figures 5.7 and 5.8. This range strikes a balance between exploration and exploitation, allowing the MCMC procedure to effectively traverse the space of Markov equivalent DAGs without getting stuck in sub-optimal regions.
Notably, in Figure 5.6, a probability of $P(MES) = 0.02$ seems to outperform all others, suggesting that the optimal probability may also be dataset-dependent. The strong performance of it on the Sachs dataset highlights the importance of tuning this parameter according to the characteristics of the data under consideration.

Further exploration can be done on dynamic tuning of this probability. For instance, it could be adjusted based on the size of the Markov equivalence class, increasing the probability as the class size grows. Naturally, this should also account for the number of nodes, as a larger node count can significantly expand the size of the equivalence class.

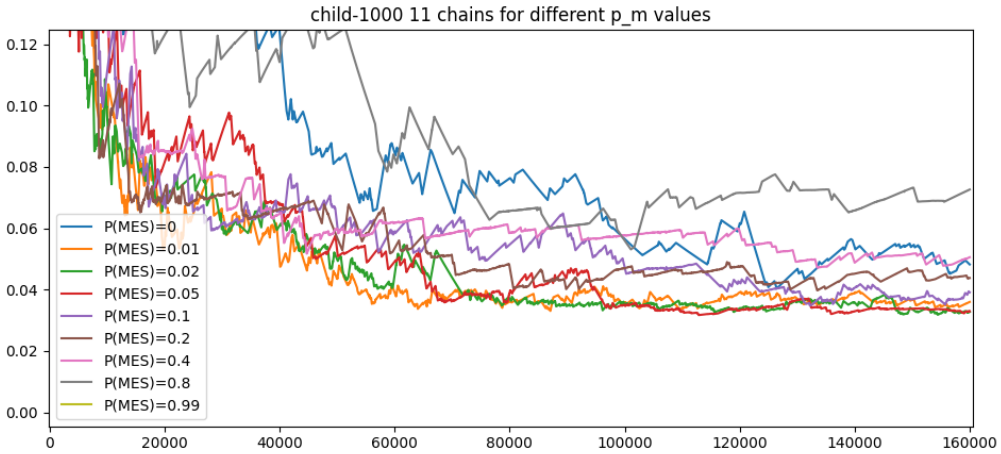

Figure 5.7: *Child* 20 variables error

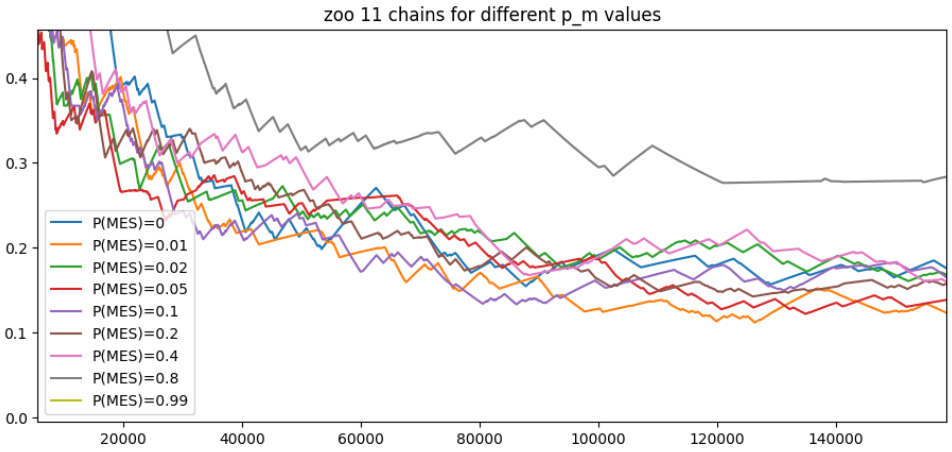

Figure 5.8: *Zoo* 17 variables error

5.3.2 Benchmark Bayesian Networks

Using the following data sets generated with max in-degree of 3 from previously mentioned benchmark Bayesian networks:

- *Sachs* - synthetic 11 variables
- *Child* - synthetic 20 variables
- *Asia* - synthetic 8 variables

For each dataset, 15 chains of each variation are generated: *Basic*, *MES*, *REV & MES* and *REV*. When chains not including REV are 10 times longer and thinned afterwards to compensate for the computationally intensive step. Figures 5.9, 5.10 and 5.11 depict side by side the median, maximum, and sum of errors of these chains from the benchmark matrix.

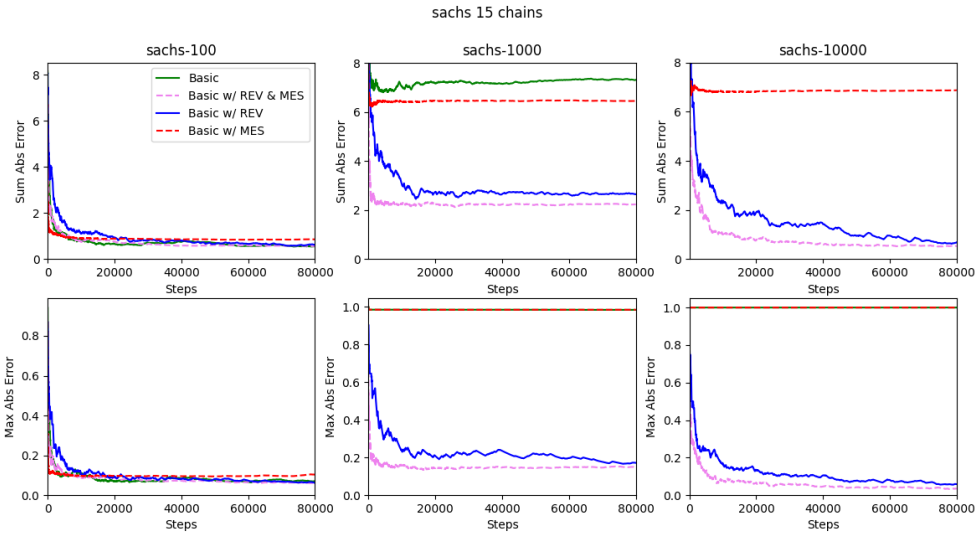

Figure 5.9: *Sachs* 11 variables error

Analysis

Examining Figures 5.9 and 5.10, specifically the 1000 and 10000 cases we see clear advantage for including a *MES* into *Basic with REV & MES*, as the error decreases more quickly. For example at sachs-10000 (Figure 5.9), it has a maximum error of 0.03, while *Basic with REV* reaches only 0.055.

Furthermore, using *MES* in isolation does not appear to provide a clear advantage. If the chain fails to progress beyond a point where an edge is consistently included, even though the reference edge probability matrix specifies a probability of 0 for that

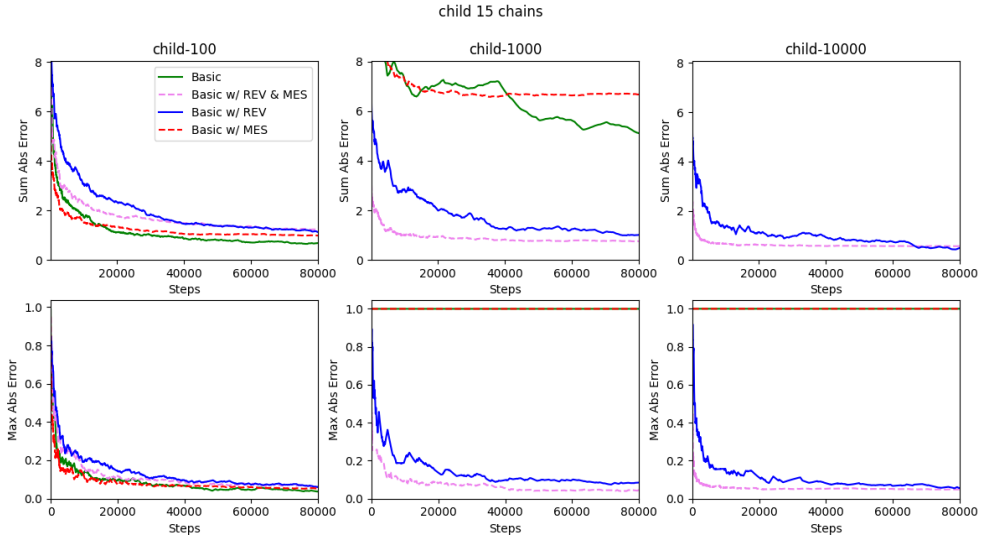

Figure 5.10: *Child* 20 variables error

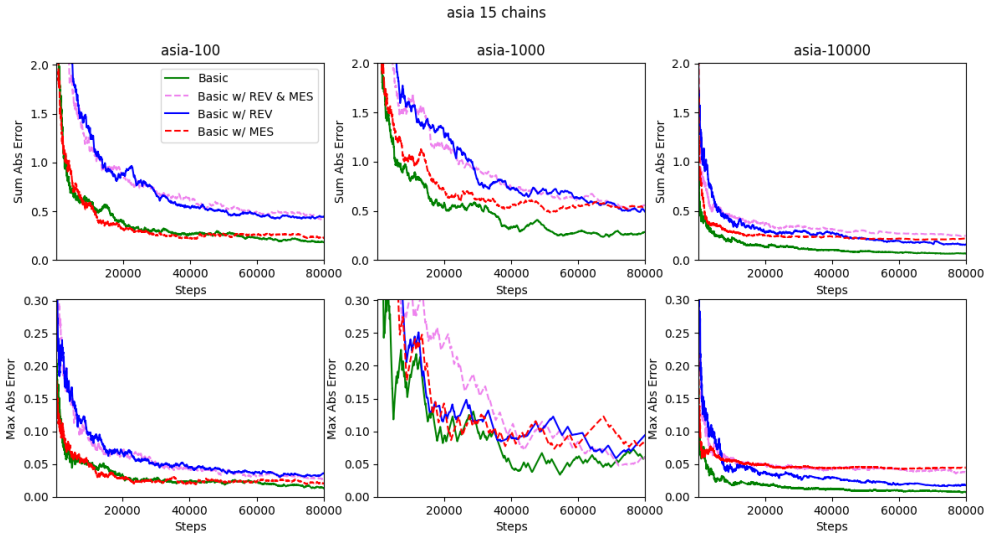

Figure 5.11: *Asia* 8 variables error

edge, sampling a *Markov equivalent DAG* will not resolve the issue, and the maximum error will remain 1.

On the other hand, *Asia* (Figure 5.11) which only has 8 variables displays a different behavior. The network appears simple enough for the basic structural chain to converge quickly to the target edge probability matrix.

5.3.3 Empirical Data Set

Experimenting further, we will try to uncover the underlying structure of a Bayesian network using the *Zoo* dataset [Forsyth, 1990], which has 17 variables. Unlike the synthetic samples generated from Bayesian networks in previous local experiments, which are sparse and limited to a maximum of 3-4 parent nodes per node—due to the computational challenges posed by large networks with too many parent sets, especially in operations like REV that require consideration of all parent sets—these samples are pre-generated and inherently different from the datasets we have used so far.

Such empirical data usually proves to be harder for *MCMC* structure discovery. Possibly, because the causal nature of variables can be interwoven and considerably more complex than constrained benchmark Bayesian networks. One notoriously difficult dataset is *agaricus-lepiota* classifying mushrooms to be poisonous or not by its features. Intuitively the nature of mushrooms properties is complex leading to difficulty in effectively discovering the underlying network. This dataset consists of too many variables for finding its exact ancestor probability matrix and therefore an alternative *zoo* dataset with less variables is used in this experiment.

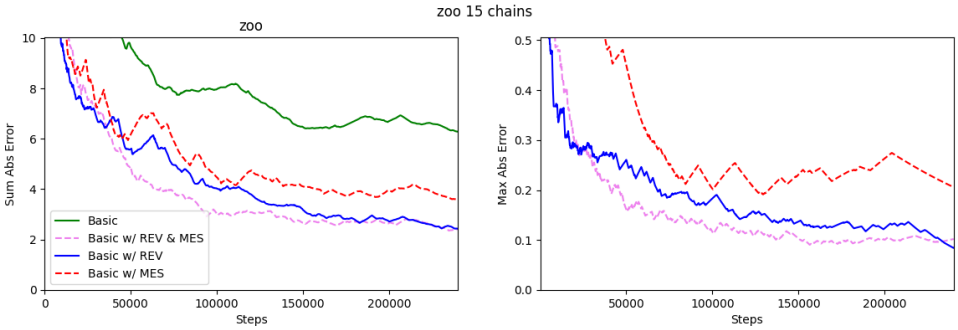

Figure 5.12: *Zoo* 17 variables error

Analysis

As clearly shown in figure 5.12, even with 300,000 steps the max absolute error gets only as low as 0.1; proving to be more difficult than previously tested data sets. Additionally, adding a *MES* to the basic 3-step and *REV* chains improves the error, yielding the best

results for most of the chain and, although ending in a similar range, strengthening our original hypothesis.

6. Partition MCMC

After analyzing trace plots and comparing edge probabilities to assess the performance of MES, we now turn our attention to Partition MCMC [Kuipers and Moffa, 2017]. This approach introduces a novel framework that leverages partition-based sampling to explore the structure space more efficiently. By focusing on partitions rather than individual edges, Partition MCMC aims to overcome limitations of traditional methods, such as slow mixing and convergence, especially in complex networks with larger variable counts. In this section, we delve into the methodology, implementation, and performance evaluation of Partition MCMC, highlighting the potential of utilizing MES for improved performance.

6.1 Background and Theoretical Framework

Partition MCMC creates a chain on the space of partitions rather than the space of DAGs. The nodes are grouped into partitions, representing a class of DAGs which can be created by setting edges from a partition "forward" only.

Partition elements are derived from a DAG by iteratively removing *outpoints*, also referred to as *sources*, which are nodes without incoming edges, and adding them as the next partition element. This process repeats until all nodes are removed, resulting in a complete partition. Demonstrated at Figure 6.1 the first outpoints are 5, 3 and 1 creating the first partition element. 4 and 2 are the second element and 6 is the third.

In the partition domain, the score of a partition is calculated by considering only the edges between nodes in different partitions, as there are no edges between nodes within the same partition. To compute the score, we sum up the scores of all possible parent sets for each node and take the product over all nodes. This is expressed as:

$$P(\mathcal{P} \mid D) \propto \prod_{i=1}^{n} \sum_{Pa_i \subseteq \mathcal{P}_{\text{right}}(X_i)} S(X_i \mid Pa_i) \tag{6.1}$$

where:

- $\mathcal{P}_{\text{right}}(X_i)$ denotes all nodes in the partition elements to the right of X_i's element.

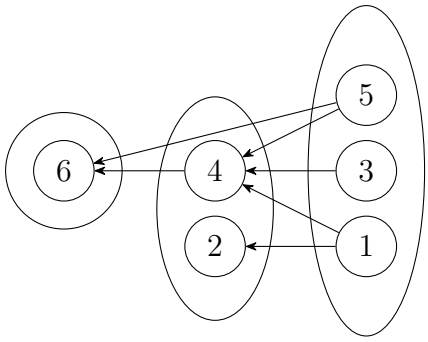

Figure 6.1: Partition of a DAG

- $\mathbf{Pa}_i$ is a valid parent set if:

$$\exists X_j \in \mathbf{Pa}_i,\ X_j \in \mathcal{P}_{\text{next}}(X_i)$$

  meaning $\mathbf{Pa}_i$ must include at least one node from $\mathcal{P}_{\text{next}}(X_i)$, the partition element immediately to the right of X_i's element.

- $S(X_i \mid Pa_i)$ is the score of X_i given its parent set $\mathbf{Pa}_i$.

Moves in a Markov chain on partitions include splitting an element or merging two adjacent ones. Splitting and joining partition elements are inverse operations, ensuring the chain's reversibility. Any partition can be reduced to a single element in at most $n-1$ steps, where n represents the number of nodes, and subsequently expanded into a new labeled partition in an additional $n-1$ steps, guaranteeing irreducibility within $2(n-1)$ steps. Additionally, by introducing a small probability of staying in the current state, the chain avoids periodicity, resulting in a proper Markov chain.

To work with DAGs from partitions—whether for estimating the original Bayesian network's probability distribution or for steps within the chain that require a DAG, such as the REV move—we sample a DAG by selecting a parent set for each node. These parent sets are chosen from the nodes in prior partitions, based on the distribution defined by their respective local likelihoods.

6.2 Experiments

Partition MCMC was implemented as a standalone runner alongside regular structural MCMC in EMCi*. It includes the capability to perform REV and MES moves and was utilized throughout these experiments.

*https://github.com/Sums-of-Products/emci/blob/main/src/partition_mcmc.py

6.2.1 Trace Plot

Previously, when dealing with trace plots, we found the Bayesian network *hailfinder* to be particularly challenging in terms of reaching DAGs with higher scores. To address this, including MES in a partition MCMC with a REV move can be proven beneficial.

To incorporate MES into a Partition Markov chain, a DAG must first be sampled from a partition. This step is computationally demanding but is already performed during a REV move. By leveraging this, each time a DAG is sampled and a REV move is executed, a MES is followed. Since the sampled equivalent DAG has the same score and an equal probability of being sampled again, the Markov chain remains symmetric. This symmetry ensures that the chain remains proper, as the MES does not introduce bias or disrupt the chain's reversibility and irreducibility.

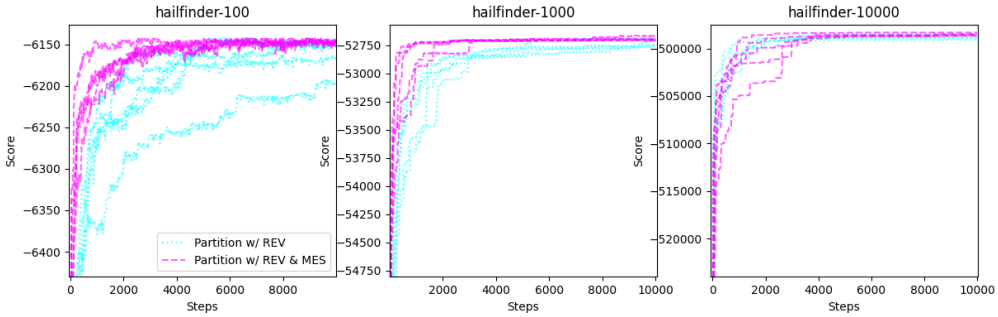

Figure 6.2: *Hailfinder* - 56 variables error

Strengthening the hypothesis, seen at the 100 and 1000 case in Figure 6.2, all 5 chains that include Markov equivalent DAG sampling reach higher scores faster compared to a chain with REV only. While a chain of 10,000 steps is relatively short, the computational intensity of partition MCMC highlights the value of adding MES. Basic chains that involve only adding, removing, and reversing edges are excluded from comparison, as they were previously shown to be inferior.

6.2.2 Edge Probability Matrix

Continuing with previously utilized benchmark, including the empirical 'zoo' we now incorporate Partition MCMC into the mix. Again, with a maximum in-degree of 3:

- *Sachs* - Synthetic 11 variables
- *Child* - Synthetic 20 variables
- *Asia* - Synthetic 8 variables
- *Zoo* - Empirical 17 variables

For each dataset, we run 15 chains of each variation: *Partition with REV*, *Partition with REV and MES*, *REV and MES* and *REV*.

The most computationally intensive aspect of Partition MCMC is sampling a DAG from the partitions; an equivalent DAG is sampled only during a REV step, with a probability of 1/15.

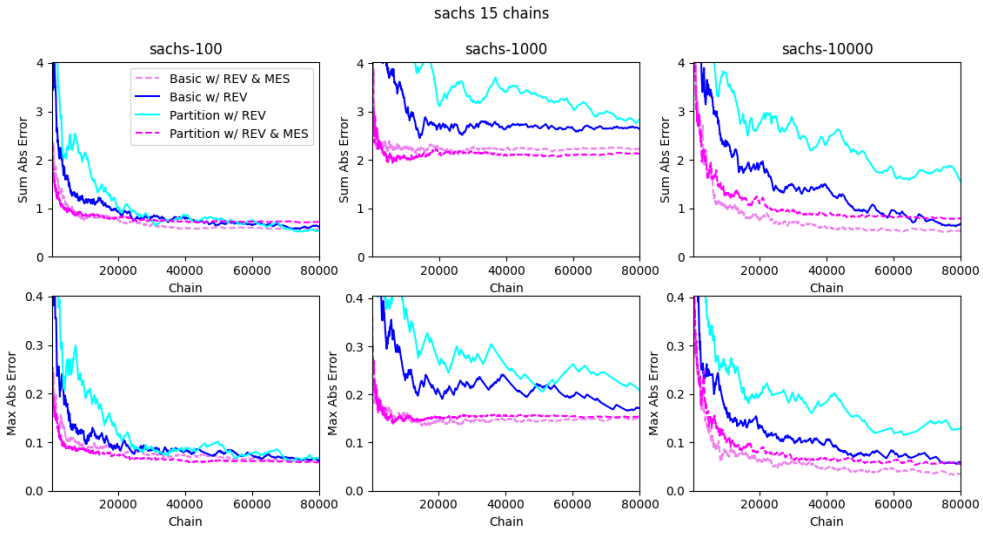

Figure 6.3: *Sachs* - 11 variables error

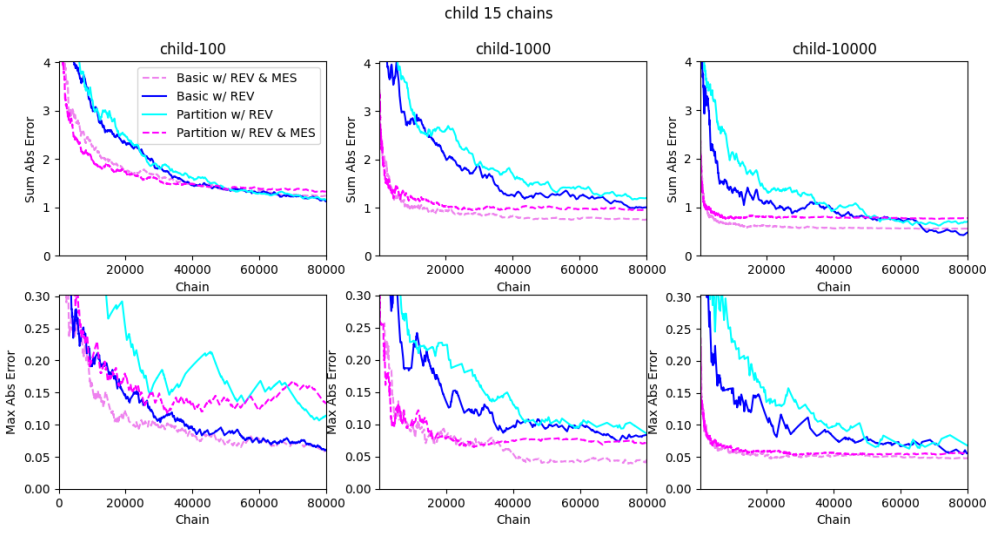

Figure 6.4: *Child* - 20 variables error

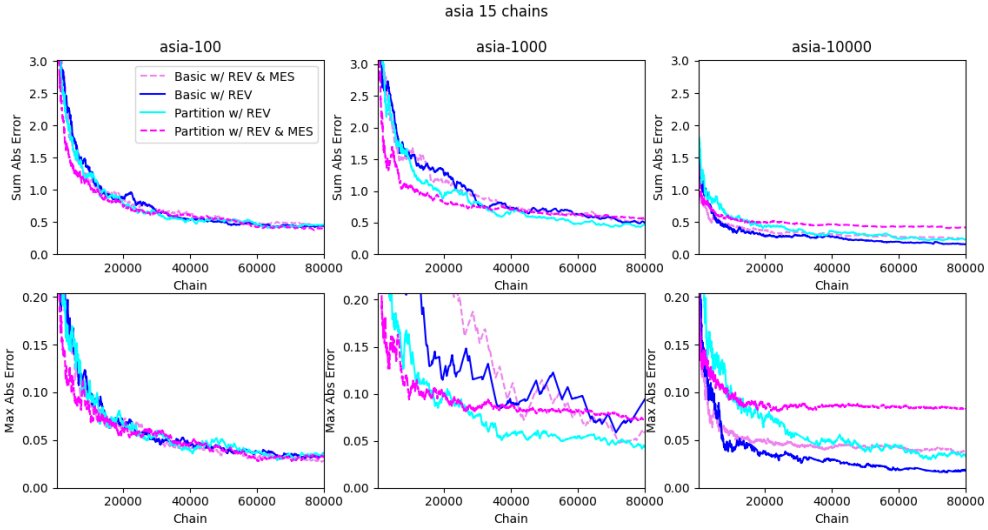

Figure 6.5: *Asia* - 8 variables error

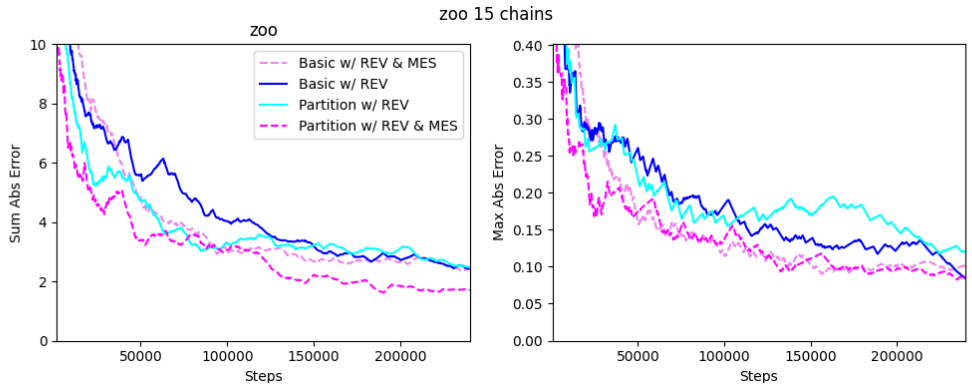

Figure 6.6: *Zoo* 17 variables error

Analysis

Once again, in Figures 6.4 and 6.3, similar to the observations with basic structural MCMC, adding a Markov Equivalence Step (*MES*) improves convergence compared to the same chain without it. Clearly, the dotted line of Partition with REV and MES reaches lower errors than Partition with just REV. This pattern is also evident for *Zoo* in Figure 6.6, which was previously shown to be non-trivial; here, Partition with REV and MES achieves the lowest sum of errors across all edges.

On the other hand, examining the results for the *Asia* network in Figure 6.5, adding MES does not improve chain convergence. This is consistent with previous findings showing that basic MCMC is sufficient for networks with a low variable count. probabilities.

Furthermore, Basic with REV and MES outperforms the Partition approach, reflecting poorly on its efficiency. Despite requiring significantly more computational resources than a basic chain, Partition MCMC fails to yield better results in these experiments. This shortcoming may stem from limited DAG diversity in the chain; a new DAG is sampled only following a REV move, which occurs on average every 15 steps. Sampling a DAG at every step is computationally intensive and does not align with the objectives of this thesis. Instead, incorporating MES proves more effective, as demonstrated by the improved convergence when comparing results with and without it.

It is crucial to emphasize that the primary focus of this thesis is not to compare basic structural MCMC to Partition MCMC. Rather, the key comparison lies within the Partition approach itself—between chains using only REV moves and those utilizing both REV and MES—highlighting the clear advantages of including MES.

7. Conclusions

Sampling a Markov equivalent DAG, termed in this thesis as the *Markov Equivalent Step* (*MES*), has been shown to improve convergence towards the target probability distribution in Bayesian network structure discovery. This method offers a practical enhancement to the performance of MCMC algorithms in addressing the challenges of this domain.

In this thesis, we explored how integrating the *Markov Equivalent Step* (MES)—a method for sampling Markov equivalent Directed Acyclic Graphs (DAGs)—into a Markov Chain Monte Carlo algorithm designed for Bayesian network structure discovery can enhance its performance.

Recent developments provided us with a polynomial-time algorithm for counting and sampling these equivalent DAGs, which we used to test our hypothesis. Sampling within the same MEC gives the MCMC the ability to make substantial "jumps" or transitions in the search space, potentially moving from local maxima to regions with better-scoring structures. We applied this approach to both structural and Partition MCMC, two widely used methods in Bayesian network discovery.

The results of our experiments provided several important insights. Incorporating MES generally improved the convergence of the chain. The most significant evidence for this came from analyzing the errors from benchmark edge probability matrices in smaller (8-20 variables) networks. While these networks are fairly easy for exact structure discovery [Parviainen and Koivisto, 2012], the findings suggest that the same advantage of MES extends to larger networks. The edge probability matrices demonstrated that MES consistently led to a faster reduction in error, indicating quicker convergence to the correct network structure. This improvement was particularly noticeable when MES was combined with the New Edge Reversal (REV) move, which is already known for helping escape local maxima. Together, MES and REV allowed for a more thorough exploration of the space of possible DAGs, leading to more accurate network structures.

A simple explanation for the faster error reduction in smaller networks, where we observed quicker convergence to the correct mode, could be that the correct mode is reached early during the runtime of the chain, and including MES helps by mixing dif-

ferent edges, leading to more accurate edge probabilities. In these cases, MES appears to facilitate the fine-tuning of the network structure by improving the accuracy of the inferred edge probabilities, which is crucial for achieving the correct model.

However, the benefits of MES were not observed in every scenario. In even smaller or simpler networks such as the 8-variables *Asia*, the advantage of MES was less pronounced, and traditional MCMC methods performed just as well without the added complexity. This suggests that while MES can be a powerful tool in the right context, it may not always be necessary, particularly in cases where the network is relatively straightforward.

Another critical aspect of our investigation was optimizing the probability of applying MES as it is essential to find the right balance. If used too frequently, the chain can end up cycling through equivalent DAGs without making meaningful progress. On the other hand, using it too sparingly might mean missing opportunities to escape local maxima. Our experiments suggested that a moderate probability of applying MES, tailored to the specific network and dataset, yields the best results.

We also explored the potential of combining MES with Partition MCMC, a computationally intensive method, particularly due to the overhead of sampling a DAG from a partition. By integrating MES into Partition MCMC, we aimed to boost its performance. The results showed that this combination improved convergence, particularly in challenging cases such as the Hailfinder network, where reaching higher-scoring DAGs is more difficult.

Nevertheless, Partition MCMC with MES did not always outperform structural MCMC with MES, particularly in smaller networks where the computational cost of Partition MCMC might outweigh its benefits. This suggests that while MES is a valuable tool, its application needs to be carefully considered in the context of the specific network and available computational resources.

In summary, this thesis contributes to the ongoing development of more efficient and robust methods for Bayesian network structure learning. By integrating MES into MCMC, we have shown that it is possible to improve the performance of these algorithms, particularly in complex and challenging scenarios. However, the effectiveness of MES is context-dependent, and its application should be tailored to the specific characteristics of the network being studied. Future research could focus on refining these techniques, perhaps by developing dynamic MES probabilities that adjust based on the performance of the MCMC chain in real-time, or by exploring the application of these methods to even larger and more intricate networks. As Bayesian networks continue to play a crucial role in probabilistic modeling, these advancements will be essential in making them more powerful and widely applicable.

Bibliography

[Andersson et al., 1997] Andersson, S. A., Madigan, D., and Perlman, M. D. (1997). A characterization of markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505–541.

[Ankan and Textor, 2023] Ankan, A. and Textor, J. (2023). pgmpy: A python toolkit for bayesian networks.

[Chickering, 1996] Chickering, D. M. (1996). *Learning Bayesian Networks is NP-Complete*, pages 121–130. Springer New York, New York, NY.

[Chickering, 2002] Chickering, D. M. (2002). Learning equivalence classes of bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498.

[Cussens, 2020] Cussens, J. (2020). Gobnilp: Learning bayesian network structure with integer programming. In Jaeger, M. and Nielsen, T. D., editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 605–608. PMLR.

[Eggeling et al., 2019] Eggeling, R., Viinikka, J., Vuoksenmaa, A., and Koivisto, M. (2019). On structure priors for learning bayesian networks. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1687–1695. PMLR.

[Elidan, 2001] Elidan, G. (2001). Bayesian Network Repository. https://www.cse.huji.ac.il/~galel/Repository/.

[Forsyth, 1990] Forsyth, R. (1990). Zoo. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5R59V.

[Friedman and Koller, 2000] Friedman, N. and Koller, D. (2000). Being bayesian about network structure. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, page 201–210, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Friedman et al., 2000] Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using bayesian networks to analyze expression data. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, RECOMB '00, page 127–135, New York, NY, USA. Association for Computing Machinery.

[Grzegorczyk and Husmeier, 2008] Grzegorczyk, M. and Husmeier, D. (2008). Improving the structure mcmc sampler for bayesian networks by introducing a new edge reversal move. *Machine Learning*, 71:265–305.

[Hastings, 1970] Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

[He et al., 2013] He, Y., Jia, J., and Yu, B. (2013). Reversible MCMC on Markov equivalence classes of sparse directed acyclic graphs. *The Annals of Statistics*, 41(4):1742 – 1779.

[He et al., 2015] He, Y., Jia, J., and Yu, B. (2015). Counting and exploring sizes of markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 16:2589–2609.

[Heckerman and Geiger, 1995] Heckerman, D. and Geiger, D. (1995). Learning bayesian networks: a unification for discrete and gaussian domains. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, page 274–284, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Jensen and Nielsen, 2007] Jensen, F. and Nielsen, T. (2007). *Bayesian Network and Decision Graphs*.

[Karan and Zola, 2016] Karan, S. and Zola, J. (2016). Exact structure learning of bayesian networks by optimal path extension. pages 48–55.

[Koivisto and Sood, 2004] Koivisto, M. and Sood, K. (2004). Exact bayesian structure discovery in bayesian networks. *J. Mach. Learn. Res.*, 5:549–573.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

[Kuipers and Moffa, 2017] Kuipers, J. and Moffa, G. (2017). Partition mcmc for inference on acyclic digraphs. *Journal of the American Statistical Association*, 112(517):282–299.

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

[Parviainen and Koivisto, 2012] Parviainen, P. and Koivisto, M. (2012). Exact structure discovery in bayesian networks with less space. *CoRR*, abs/1205.2620.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[Pensar et al., 2018] Pensar, J., Kohonen, J., and Corander, J. (2018). Exact bayesian learning of partition directed acyclic graphs. *Journal of Physics: Conference Series*, 1036:012017.

[Pensar et al., 2020] Pensar, J., Talvitie, T., Hyttinen, A., and Koivisto, M. (2020). A bayesian approach for estimating causal effects from observational data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5395–5402.

[Schwarz, 1978] Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461 – 464.

[Talvitie et al., 2018] Talvitie, T., Eggeling, R., and Koivisto, M. (2018). Finding optimal bayesian networks with local structure.

[Talvitie and Koivisto, 2019] Talvitie, T. and Koivisto, M. (2019). Counting and sampling markov equivalent directed acyclic graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7984–7991.

[Talvitie et al., 2020] Talvitie, T., Vuoksenmaa, A., and Koivisto, M. (2020). Exact sampling of directed acyclic graphs from modular distributions. In Adams, R. P. and Gogate, V., editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 965–974. PMLR.

[Wienöbst et al., 2021] Wienöbst, M., Bannach, M., and Liskiewicz, M. (2021). Polynomial-time algorithms for counting and sampling markov equivalent dags. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12198–12206.