2016

Using node ordering to improve Structure MCMC for Bayesian Model Averaging

Abhineet Sharma
Iowa State University

Recommended Citation

Sharma, Abhineet, "Using node ordering to improve Structure MCMC for Bayesian Model Averaging" (2016). *Graduate Theses and Dissertations*. 16014.
https://lib.dr.iastate.edu/etd/16014

Using node ordering to improve Structure MCMC for Bayesian Model Averaging

by

Abhineet Sharma

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:

Jin Tian, Major Professor

Kris de Brabanter

Xiaoqiu Huang

Iowa State University

Ames, Iowa

2016

DEDICATION

I would like to dedicate this thesis to my family and my friends without whose support and loving guidance I would not have been able to complete this work.

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

DAG(s)	Directed Acyclic Graph(s)
MCMC	Markov Chain Monte Carlo
$\mathbb{X}$	Set of random variables
$\mathbb{V}$	Set of vertices of DAG. i representing i^{th} vertex
$\mathbb{E}$	Set of edges of DAG
$i \to j$	an edge from node i to j
$Pa_G(X_i)$	set of parents of node i in DAG, G
$\mathbb{S}$	Set of DAG structures (structure space)
$\mathcal{G}$	Set of sampled DAGs
$\mathcal{B}$	a Bayesian network(DAG & joint distribution combined)
Θ_G	joint probability distribution of nodes in G
$\prec$	Node Ordering specifying the nodes preceding other nodes
$G_\prec$	DAG consistent with an order $\prec$
MH-Algorithm	Metropolis Hastings Algorithm
i.i.d.	Independent, Identically Distributed

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Jin Tian for his insight and substantial contribution to this research and his guidance and support throughout my time at Iowa State University. He had been a strong pillar of support throughout my research. His kindness and valuable inputs made my work pleasant and enjoyable. I would like to sincerely express my heartfelt gratitude as otherwise this work would not been possible.

I would also like to thank my committee members for their efforts and contributions to this work: Dr. Kris de Brabanter and Dr. Xiaoqiu Huang. They have been very helpful and approachable throughout the course of this research. Their expertise in the respective fields is a motivation to venture deep into the research world.

I would also like to extend my gratitude towards Dr. Alexander Stoytchev for his valuable encouragement, suggestions and insight. He is a great teacher and a motivation. I would also like to thank Dr. Mathew Frank for providing me an opportunity to work as a research assistant in Rapid Manufacturing and Prototyping lab headed by him. It was a great learning experience that helped fund my graduate program and a source of some of very interesting problems that I got a chance to work on.

My expression of gratitude remains incomplete without the mention of Department of Computer Science and Iowa State University for providing a platform for conducting research and development. I will always remain grateful for this opportunity.

Finally, I am eternally grateful to my family and all my friends who made my stay at Iowa State University a wonderful experience. I am indebted to them for their support, patience and love throughout this journey. It would not have been any easier without them.

ABSTRACT

In this thesis, I address an important problem of estimating the structure of Bayesian network models using Bayesian model averaging approach. Bayesian networks are probabilistic graphical models which are widely used for probabilistic inference and causal modeling. Learning the structure of Bayesian networks can reveal insights into the causal structure of the underlying domain. Owing to the super exponential structure space, it is a challenging task to find the most suitable network model that explains the data. The problem is worsened when the amount of available data is modest, as there might be numerous models with non negligible posterior. Therefore, we are interested in the calculation of posterior of a feature like presence of an edge from one particular node to another or a particular set being a parent of a specific node. The contribution of this thesis includes a Markov Chain Monte Carlo simulation approach to sample network structures from a posterior and then using Bayesian model averaging approach to estimate the posterior of various features.

CHAPTER 1. INTRODUCTION & MOTIVATION

In this chapter, we present the problem of learning network structure from data and the motivation behind this problem. The chapter is structured as follows. Section 1.1 provides description of Bayesian network models in section 1.1.1 followed by Bayesian approach to learning Bayesian networks in section 1.1.2. Section 1.2 defines the problem our proposed approach tries to address along with the motivation behind the problem. Section 1.3 presents the organization of the remainder of the thesis.

1.1 Introduction

In order to understand the problem itself and the challenges posed by learning the underlying Bayesian network model structure from data, it is important to understand Bayesian network models and the Bayesian approach towards the probabilistic estimation of network structures to explain the data.

1.1.1 Bayesian Networks

A Bayesian network is a graphical model that encodes probabilistic relationships amongst variables of interest. Over a period of time, Bayesian networks have become a popular representation for recording knowledge in expert systems. Bayesian networks provide the following advantages. Firstly, because they encode both causal and probabilistic semantics, it is an ideal representation for combining the prior knowledge(which is in causal form) with the data. Secondly, they are helpful in learning about the underlying domain especially in the cases where there is a limited or no prior knowledge available. Thirdly, Bayesian networks render themselves very useful in case of incomplete data sets. For example, consider a classification or regression problem where two variables(say) are strongly correlated. In case of complete data, correlation is not an issue with standard supervised learning approaches. But, when one of the input is not observed, most of the models would produce inaccurate results because they do not encode correlation between those two variables. Bayesian networks, on the contrary, provide a natural way to encode such dependencies. Lastly, when combined with Bayesian methods, there is no need to hold out some data for testing as entire data available can be used for training the model. Moreover, the acyclic nature of the graph(underlying representational model) and its edges representing the conditional independence relationships amongst variables, it makes it very easy to represent the joint distribution.

A Bayesian network, $\mathcal{B}$, for a set of variables $\mathbb{X} = \{X_1, X_2, \ldots, X_N\}$ consists of (1.) a Directed Acyclic Graph(DAG), $G = \{\mathbb{V}, \mathbb{E}\}$ that encodes set of conditional independence relations amongst variables in $\mathbb{X}$ (2.) a set Θ_G of local probability distributions associated with each variable. Nodes in G are in one-to-one correspondence with the variables in $\mathbb{X}$. For the sake of convenience, let use the index set, $\mathbb{V} = \{1, 2, \ldots, N\}$ to represent a variable X_i with an index i. Additionally, we will use the index i synonymously for a node in a graph as well as the random variable X_i as they both represent the same entity.

The set of edges $\mathbb{E} = \{(i, j) : \exists \text{ an edge from } X_i \rightarrow X_j \text{ in } G\}$ represents the causal relationships amongst the variables in $\mathbb{X}$. For the sake of convenience, we will use the notation $i \rightarrow j$ to represent an edge from node i to node j, which signifies that X_i is parent of X_j. It also represents the cause-effect relationship between i and j, i being the cause. Since an edge encodes conditional independence relationship between two variables, given all parents of a node i, it is independent of all other variables in $\mathbb{X}$. If we denote, parent set of a variable X_i in a DAG G as $Pa_G(X_i)$, then by the nature of construction of DAG G, we have the joint probability distribution for $\mathbb{X}$ represented by G as,

$$P_G(\mathbb{X}) = \prod_{i=1}^{N} p(X_i | Pa_G(X_i)) \tag{1.1}$$

Let us conclude this section with an example of Bayesian network from Heckerman (1996) concerning the problem of predicting whether or not a credit card transaction is fraudulent.

Let us use the following choice of variables: (1.) Fraud(F), representing if there is a fraud or not (2.) Gas(G), representing if a purchase was made at a gas station or not (3.) Jewelery(J), representing if jewelery was purchased by credit card or not (4.) Age(A), representing age of the card holder (5.) Sex(S), representing gender of the card holder. One of the possible representations of Bayesian network representing this model could be as referred in figure 1.1

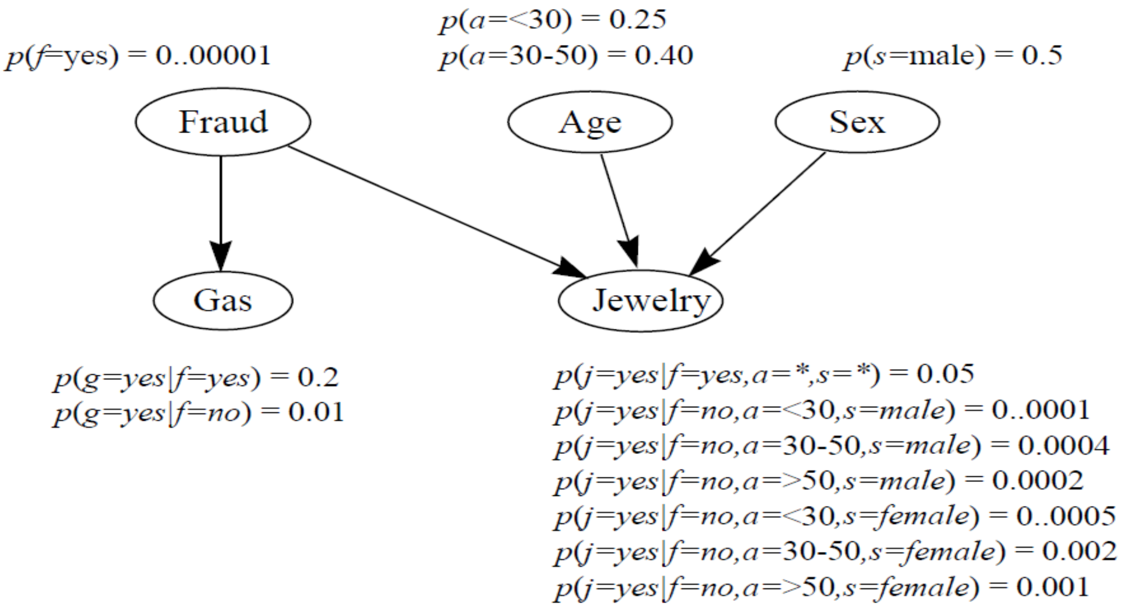

p(a=<30) = 0.25
p(a=30-50) = 0.40
p(f=yes) = 0..00001
p(s=male) = 0.5
Fraud
Age
Sex
Gas
Jewelry
p(g=yes|f=yes) = 0.2
p(g=yes|f=no) = 0.01
p(j=yes|f=yes,a=*,s=*) = 0.05
p(j=yes|f=no,a=<30,s=male) = 0..0001
p(j=yes|f=no,a=30-50,s=male) = 0.0004
p(j=yes|f=no,a=>50,s=male) = 0.0002
p(j=yes|f=no,a=<30,s=female) = 0..0005
p(j=yes|f=no,a=30-50,s=female) = 0.002
p(j=yes|f=no,a=>50,s=female) = 0.001

Figure 1.1: A possible Bayesian network representing model for detecting credit card fraud. *Source:* Heckerman (1996)

From the representation in figure 1.1, if we consider the following ordering of variables in the chain rule (F, A, S, G, J). We have

$$\begin{aligned} P(A|F) &= \quad P(A) \\ P(S|F,A) &= \quad P(S) \\ P(G|F,A,S) &= \quad P(G|F) \\ P(J|F,A,S,G) &= \quad P(J|F,A,S) \end{aligned}$$

$$\therefore P(F,A,S,G,J) = P(F)P(A)P(S)P(G|F)P(J|F,A,S)$$

Hence, in order to infer if there was a fraud, given the values of all other variables, we can use *Bayes' rule*

$$P(f|a,s,g,j) = \frac{P(f,a,s,g,j)}{\sum_{f'} P(f',a,s,g,j)}$$

There have been a lot of work on various inference algorithms that exploit the simplification of joint distribution by the use of Bayesian networks. However, discussion on this is out of the scope of this document. Heckerman (1996) talks at length about Bayesian networks and some of the initial inference approaches towards inference and learning. We have laid enough groundwork to understand Bayesian networks. To further understand the problem, we need to look at the Bayesian approach towards learning given data, which is presented in the next section.

1.1.2 Bayesian Model Averaging approach to learning Bayesian networks

Consider the problem of analyzing the distribution over some set of random variables $\mathbb{X} = \{X_1, X_2, \ldots, X_N\}$ based on a fully observed data set with M observations, $D = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^M\}$, with each $\mathbf{x}^j$ being complete assignment to the variables in $\mathbb{X}$. It must be noted that in the remainder of the document, we only talk about the case of *complete data*, that is, all the instantiations of random variables are known in the data set and there is no missing data. In Bayesian approach to learn Bayesian networks from a training data set D, we compute the posterior probability of a network, G, using Bayes' rule

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

We then compute the posterior probability of any hypothesis of interest by averaging over all possible networks. If we are interested in some feature $f(G)$, for example presence of an edge $i \to j$ in G, defined as follows

$$f(G) = \begin{cases} 1, \exists \text{ an edge } i \to j \text{ in } G \\ 0, otherwise \end{cases} \tag{1.2}$$

Then we have,

$$P(f|D) = \sum_{\mathcal{G}} f(G) P(G|D) \tag{1.3}$$

Bayesian paradigm suggests to start by defining a prior probability distribution $P(\mathcal{B})$ over the space of possible Bayesian networks $\mathcal{B}$. This prior is then updated using Bayesian conditioning to give a posterior distribution $P(\mathcal{B}|D)$. Recall from section 1.1.1, the description of a model $\mathcal{B}$ has two components (1.) DAG, G (2.) Numerical parameters, Θ_G that define a multinomial distribution $\theta_{X_i|Pa_G(X_i)}$. Therefore, in order to define the prior $P(\mathcal{B})$, we need to define discrete probability distribution over the graph structures in structure space, $\mathcal{S}$ and a local probability distribution for all the variables for each graph $G \in \mathcal{S}$. Given sufficient amount of data in comparison to the number of variables being observed, prior over structures loses its significance because of Bayesian conditioning of the parameters. Therefore, a simple prior can be chosen such as uniform structure prior (Heckerman, 1996). There exist other priors as well that assign greater probability to dense networks (Buntine, 1991). We use an alternative prior which considers the number of possibilities of determining parents of each node. There are $\binom{n-1}{k}$ possible parent sets given that a node X_i has k parents. Thus,

$$P(G) \propto \prod_{i=1}^{N} \binom{n-1}{|Pa_G(X_i)|}^{-1} \tag{1.4}$$

All the priors have a common property called *Structure modularity*, which implies that choice of families for different nodes is independent a priori. Thus, prior can be decomposed as follows.

$$P(G) = \prod_{i=1}^{N} Q_i(Pa_G(X_i)) \tag{1.5}$$

Thus, we have

$$Q_i(Pa_G(X_i)) = \binom{n-1}{|Pa_G(X_i)|}^{-1} \tag{1.6}$$

Furthermore, we assume

- *Global parameter independence*, $P(\Theta_G|G) = \prod_i P(\theta_{X_i|Pa_G(X_i)}|G)$
- *Parameter Modularity* For any two graphs, G and G', let $Pa_G(X_i) = Pa_{G'}(X_i) = \mathbf{U}$, then $P(\theta_{X_i|U}|G) = P(\theta_{X_i|U}|G')$

Selection of parameter prior depends upon the types of parametric families we consider. For discrete networks, we use the standard assumption of *Dirichlet* prior (Heckerman, 1996) and for Gaussian networks, we use *Wishart* prior.

Under the assumptions, (1.) Structure modularity (2.) Global parameter independence (3.) Parameter modularity the marginal likelihood of data $P(D|G)$ can be decomposed into the product of local marginal likelihoods, often called as local scores (Cooper and Herskovits, 1992; Heckerman, 1996).

$$P(D|G) = \prod_{i=1}^{N} score_i(Pa_G(X_i)|D) \tag{1.7}$$

where, $score_i(Pa_G(X_i)|D)$ represents the score of node i given data. Using parameter priors that satisfy the above mentioned assumptions, it can be shown that $score_i(Pa_G(X_i))$ has a closed form solution. Thus, $P(G|D)$ can be evaluated using Bayes' rule,

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

To conclude, Bayesian model averaging to learn Bayesian networks includes defining a feature we are interested in and then averaging over all the graphs to estimate our belief in the existence of the feature in the true model. The approach includes specifying a prior distribution for all the graphs in the structure space and then calculating the parameters specifying the behavior of nodes for all of the graphs using the data to obtain a joint distribution for all the Bayesian networks. Under certain assumptions, this approach provides a closed form solution. Evaluation of parameters given data involves solving complex integrals which brings some estimation techniques into picture. Markov Chain Monte Carlo(MCMC) simulation is one of the most popular techniques, which we use in order to propose a solution to our problem. Additionally, the structure space being super-exponential further worsens the problems. We address our problem and the challenges involved in solving the problem in the next section.

1.2 Problem & Motivation

In this section, we address the problem of learning the underlying true model for a particular domain. In certain cases, the goal to learn the model is not for prediction purposes but for discovering the domain structure. One very good example for this particular application is the *"Gene expression data"*, where we might want to use Bayesian learning to understand the mechanisms by which genes in a cell produce proteins, which in turn cause other genes to express themselves or prevent them from doing so. Thus, our main goal is to understand the causal relationships between the expression levels of various genes. In this case, we have a very limited or no prior understanding of the domain and a good estimation about the underlying model would be very useful.

Structure learning is considered to be a hard problem owing to the super-exponential $\left[O\left(n!2^{\frac{n(n-1)}{2}}\right)\right]$ structure space. Many approaches have been proposed in order to tackle this problem and reach good estimation results. Broadly, all the approaches can be divided as follows. (1.) Constraint Based (2.) Score and Search.

The edges of Bayesian network encode conditional dependence relationships, which forms the basis of constraint based learning methods. Starting from a complete skeleton, decisions about whether or not to keep the edges are made via conditional independence tests. Starting from pairwise comparisons they might extend increase the complexity. Such tests are sensitive to local errors and the order in which they are run.

Score and search algorithms rely on a measure of fit to score each and every model and search for the highest scoring model. Heckerman (1995) has shown that a high scoring model is more likely to occur by a substantial amount than any other model. In certain cases this might not hold and there might be many structures with non-negligible posteriors. This is because the amount of data available is modest in comparison to the number of variables involved. For example, for gene expression data, there are a few hundred experiments that collect expression data for thousands of genes. So, it is highly likely that there are more number of models that explain the data reasonably well. Hence, we can not be confident on the true representation of the underlying model. However, we can extract certain features of the domain reliably. For instance, if two variables are strongly correlated it is highly likely that there exists an edge between those two variables in most of the high scoring structures. Thus, we can estimate such features using Bayesian model averaging approach as has already been discussed in section 1.1.2.

In this section, we presented the problem and the challenges posed by this problem. Our approach uses score and search technique to sample enough structures from the posterior distribution $P(G|D)$ using MCMC sampling technique and then use Bayesian model averaging to provide an estimate for certain features we might be interested in. Before presenting our approach, we discuss some of the proposed methods in chapter 3 in detail. Next section provides an outline of the remainder of the document.

1.3 Thesis Outline

In this section we have provided the groundwork for Bayesian model averaging technique. The remainder of this thesis is organized as follows. In chapter 2, we talk about Markov Chain Monte Carlo simulation technique that forms the basis of our approach and all the other approaches we discuss in detail. Chapter 3 focuses on some of the approaches that have been proposed in order to tackle this problem. Then we present our approach in chapter 4 followed by experimental setup and results in chapter 5. It is followed by chapter 6 which provides analysis of the results in comparison to other methods and presents some of the improvements that can be made to this approach. Finally, this document is concluded by all the bibliographic references cited in this document.

CHAPTER 2. MARKOV CHAIN MONTE CARLO(MCMC)

In this chapter, we present a very powerful sampling technique called Markov Chain Monte Carlo(MCMC). This technique forms the basis of our method. This section covers only the relevant details providing introduction to MCMC in section 2.1 followed by theory behind in MCMC methods in section 2.2. We then present a very popular sampling algorithm called Metropolis-Hastings(MH) in section 2.3. We have used MH algorithm in all the experiments for our approach. Finally, we conclude this chapter with a brief summary.

2.1 Introduction & Motivation

Metropolis Hastings algorithm is a widely used procedure for sampling from a specified distribution on a large finite set. This algorithm is an instance of a large class of sampling algorithms known as Markov Chain Monte Carlo(MCMC). These algorithms have played a significant role in various domains such as statistics, econometrics, physics and computer science. One of the main reasons for the popularity of MCMC methods is their ability to estimate complex integrals which might be difficult to solve analytically. In 1946, while recuperating from an illness, Stan Ulman was trying to compute the chances that a particular solitaire laid out with 52 cards would come out successfully. After attempting many exhaustive calculations, he tried to solve the problem practically by laying out various solitaires at random and observing and counting successful plays. This idea of selecting a statistical sample to approximate a hard combinatorial problem by a much simpler problem is at the heart of Monte Carlo simulations (Andrieu et al., 2003). For numerical problems in a large number of dimensions, Monte Carlo methods are often more efficient than conventional numerical methods. However, implementation of Monte Carlo methods requires sampling from a high dimensional probability distributions and this may be very difficult and expensive in analysis and computation time. We discuss the theory behind MCMC methods in the next section.

2.2 Theory

MCMC techniques are often applied to solve integration and optimization problems in large dimensional spaces. These two types of problems play a fundamental role in machine learning, physics, statistics, econometrics and decision analysis. MCMC is a general purpose technique to draw $i.i.d$(independent, identically distributed) samples from a target probability distribution in an N-dimensional space $\mathcal{X}$, using random numbers drawn from uniform probability in certain range. We then use these samples along with the target stationary distribution to estimate some value of interest. To further understand it, let us take an example of the problem of Bayesian inference (Gilks et al., 51 1) because of the nature of the problem addressed by this thesis.

2.2.1 Problem of Bayesian Inference

From Bayesian perspective, there is no fundamental distinction between observables and parameters of a statistical model and all are considered random quantities. Let D denote the data and θ the parameters. For formal inference, we need to set up a joint distribution model $P(D, \theta)$ over all random quantities. We have,

$$P(D, \theta) = P(D|\theta)P(\theta)$$

Having observed D, we can use Bayes' theorem to determine the posterior distribution of θ, $P(\theta|D)$.

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{\int P(\theta)P(D|\theta)d\theta}$$

All the quantities that can be expressed in the terms of posterior expectations of functions of θ can be calculated as follows.

$$E\left[f(\theta)|D\right)] = \frac{\int f(\theta)P(\theta)P(D|\theta)d\theta}{\int P(\theta)P(D|\theta)d\theta}$$

Integrals of this sort are not easy to compute analytically especially when the dimensionality increases, which is the case in most of the applications involving Bayesian inference. In such cases, MCMC is a very powerful method. There exist some alternate numerical approaches as well which become difficult and inaccurate at higher dimensions.

In order to focus solely on MCMC, let us express the same problem in more general terms. Let X be a vector with k random variables with distribution $\pi(.)$(posterior of graph given data, in our case). The task is to evaluate the expectation, $E[f(X)]$ as follows.

$$E\left[f(X)\right] = \frac{\int f(x)\pi(x)dx}{\int \pi(x)dx}$$

where, $f(x)$ is the function of interest(evaluating presence of an edge in structures). Here, we allow the possibility that the distribution of X is known only up to a constant of normalization i.e. $\int \pi(x)dx$ is unknown. Thus, for the Bayesian inference case, we know that $P(\theta|D) \propto P(\theta)P(D|\theta)$.

Having defined the problem, we now introduce MCMC in the next section.

2.2.2 MCMC

MCMC consists of two parts (1) Monte Carlo Integration (2) Markov Chain. In order to completely understand the simulation methodology, let us first understand these concepts.

Monte Carlo Integration

Monte Carlo integration evaluates $E[f(X)]$ by drawing samples $\mathbb{X} = \{X_1, X_2, \ldots, X_N\}$ from $\pi(.)$ and then approximating the value of interest as follows.

$$E\left[f(X)\right] \approx \frac{1}{N}\sum_{X\in\mathbb{X}} f(X) \tag{2.1}$$

When the samples are independent, law of large numbers ensures that the approximation can be made as accurate as desired by increasing the sample size N. In general, drawing samples independently from $\pi(.)$ is not always feasible. However, the samples need not necessarily be independent. Using Markov chains that have $\pi(.)$ as stationary distribution, we can draw such samples.

Markov Chain

A Markov chain is a stochastic process that fulfills *Markov assumption*:

$$P(X_{i+1} = x_{i+1}|X_0 = x_0, X_1 = x_1, \ldots, X_i = x_i) = P(X_{i+1} = x_{i+1}|X_i = x_i) \tag{2.2}$$

It implies that the probability of the system being in a state x_{i+1} at time $t = i+1$ is conditionally dependent only upon its previous state at time $t = i$ and no other states before that. Thus a Markov chain can be specified by (1) the initial distribution, $\pi(x_0)$ (2) transition kernel $q(x_{i+1}|x_i)$.

The idea behind MCMC methods is to start with an initial state $X_0 = x_0$. This initial state could be random of a fixed one as we shall see that after some time the chain forgets about its initial state and converges to the stationary target distribution $\pi(.)$. Upon convergence all subsequent steps are draws from the stationary distribution $\pi(.)$. All the MCMC methods are based on the same idea with the only difference in how the transitions in the Markov chain are created. Under certain conditions, the distribution $\pi^{(n)}$ will tend towards a unique stationary state as $n \to \infty$. More specifically, convergence requires that Markov chain is

- *Irreducible*: the state space can not be partitioned into non-communicating subsets. It implies that it is possible to move from each state to every other state in a finite number of moves.
- *Positive Recurrent*: if the chain is run long enough, it will return to its initial state with a probability 1.
- *Aperiodic*: a Markov chain is periodic if it is only possible to transition from state i to itself in multiples of N steps, where $N > 1$.

For finite state spaces, irreducibility implies positive recurrence, and it is usually easy to ensure that the state space is irreducible. Periodicity is slightly more subtle. There is another requirement at convergence of the chain, *global balance*, which introduces us to the concept of *reversibility* described as follows.

Reversibility

When the chain converges, the distribution $\pi(.)$ satisfies the equation $\pi(j)q(i|j) = \pi(i)$. This implies a series of equalities of the form

$$\sum_j \pi(j)q(i|j) = \pi(i), \forall i \tag{2.3}$$

Since $\sum_j q(j|i) = 1$,

$$\begin{aligned} \sum_j \pi(j)q(i|j) = & \quad \pi(i)\sum_j q(j|i) \\ = & \quad \sum_j \pi(i)q(j|i), \forall i \end{aligned} \tag{2.4}$$

Equation 2.4 is called *global balance*, whereby the probability of moving into a state equals the probability of leaving the state, which is an obvious requirement at convergence. One way to ensure global balance is to stipulate a stronger condition of *detailed balance*, also known as *reversibility*, which states that

$$\pi(j)q(i|j) = \pi(i)q(j|i) \tag{2.5}$$

Summing both sides over j, we can see that equation 2.5 implies global balance. It must be noted that although reversibility implies global balance, it is not a necessary condition for global balance. There is a possibility that the chain might have a tendency to sway towards either side, thereby affecting reversibility. To counter this, some of the MCMC algorithms introduce *acceptance probability* as a balancing factor to ensure reversibility, as we shall see in section 2.3 in Metropolis-Hastings algorithm. It is usually denoted by α defined as follows.

$$\begin{aligned} \alpha(i|j)\pi(j)q(i|j) = & \quad \pi(i)q(j|i) \\ \implies \alpha(i|j) = & \quad \frac{\pi(i)q(j|i)}{\pi(j)q(i|j)} \end{aligned}$$

Since, $\alpha(i|j)$ is the probability of accepting the move from state j to state i, we have

$$\alpha(i|j) = min\left(1, \frac{\pi(i)q(j|i)}{\pi(j)q(i|j)}\right) \tag{2.6}$$

Having covered most of the theory behind MCMC methods, before presenting MH algorithm, we present the concepts of ***burn-in*** and ***thinning*** as follows.

Auto-correlation, burn-in & thinning

In a Markov chain, given X_{n-1}, X_n is independent of all the previous observations. However, since X_n depends upon its predecessor, this induces a non-zero correlation between X_n and X_{n+d} , even when the distance $d > 1$. The correlation between X_n and X_{n+d} is termed as *auto-correlation* at lag d, and for a Markov chain that converges to a unique stationary distribution we expect the auto-correlation to decrease as the lag is increased. So, we let the chain *burn-in* for some iterations and discard them before we start gathering samples. Thus, after a subsequently long ***burn-in*** time of say m iterations, points $\{X_t : t = m + 1, \ldots, n\}$ will be dependent samples approximately from our target stationary distribution $\pi(.)$. To further obtain i.i.d. samples, we introduce the idea of ***thinning*** of chain, where upon obtaining each subsequent sample we let the chain run for some iterations in order to obtain the next sample which is independent of the previous one.

We now present one of the most popular MCMC algorithms, Metropolis-Hastings algorithm in the next section.

2.3 Metropolis-Hastings Algorithm

Metropolis-Hastings algorithm follows a rejection sampling approach where values are drawn from an arbitrary(yet sensibly chosen) distribution and 'corrected' so that asymptotically they behave as random observations from the target distribution. Metropolis-Hastings was first described by Hastings (1970) as a generalization of the Metropolis algorithm of Metropolis et al. (1953). MH-Algorithm starts with a random initial state, X_0. Then, at time t it proposes a move to another state X_{t+1} from the current state X_t using the transition probability $q(X_{t+1}|X_t)$. Then this proposal is accepted with an acceptance probability,

$$\alpha(X_{t+1}|X_t) = min\left(1, \frac{\pi(X_t)q(X_{t+1}|X_t)}{\pi(X_{t+1})q(X_t|X_{t+1})}\right) \tag{2.7}$$

If the proposed move is rejected, the chain remains at the current state. We now present MH-Algorithm.

Metropolis-Hastings Algorithm

I. Initialize X_0 and $t \leftarrow 0$

II. Repeat,

   (i) Sample a point X_{t+1} from $q(X_{t+1}|X_t)$

   (ii) Evaluate acceptance probability of the proposed move $\alpha(X_{t+1}|X_t)$ as specified in equation 2.7

   (iii) Sample $u \sim U[0, 1]$

   (iv) If $u > \alpha(X_{t+1}|X_t)$, then $X_{t+1} \leftarrow X_t$

   (v) $t \leftarrow t + 1$

This algorithm defines random walk across the Markov chain. We can use this to gather samples by using appropriate *burn-in* and *thinning* iterations. We now present MH-Algorithm in context with our method, Combined DAG Order MCMC in the next section.

For our method, we use MH-Algorithm to sample DAGs from a target stationary distribution $P(G|D)$ and then use the same to estimate two specific features of the true underlying model (i) $P(i \to j|D)$, the probability that there exists a causal relationship between i being the cause and j being the effect. (ii) $P(Pa_i = U|D)$, the probability that all the elements of a given set U have a causal relationship with i being the effect.

To summarize, Markov Chain Monte Carlo methods provide a powerful way to estimate hard combinatorial problems which are difficult to solve analytically by providing a simpler way to calculate complicated integrals. MCMC methods are widely used in solving integration and optimization problems in high dimensional domains, Bayesian inference being one of them. Under certain conditions, we can draw i.i.d. samples from a specific target distribution by defining an appropriate Markov chain and using a random walk algorithm to move around the chain and gather samples. Thus, MCMC provides a very powerful simulation technique for various domains. We then presented one of the most popular of MCMC methods, Metropolis-Hastings algorithm followed by an overview of using it as a vessel to sample graph structures to solve our problem. We have set enough ground work to present our method. Before presenting our method, we present some of the previous work that has been carried out in regards to this problem in the next section.

CHAPTER 3. PREVIOUS WORK

In the last few decades there has been a lot of research focused on the problem of learning Bayesian networks from data (Buntine, 1991; Heckerman, 1996). The key question is to learn the relationship between interacting entities and their surrounding environments(1.2) for example *gene regulatory networks* (Friedman et al., 2000; Husmeier, 2003; Friedman, 2004; Rau et al., 2012) and *cellular signaling networks* (Sachs et al., 2005; Mukherjee and Speed, 2008; Hill et al., 2012). Probabilistic graphical models provide a framework for characterizing the joint probability distribution of the variables in a domain and making inferences about features of interest. Bayesian networks are a popular class of probabilistic graphical models. Daly et al. (2011) provide a comprehensive overview of approaches for Bayesian networks learning. Due to the super-exponential structure space, structure learning is considered a hard problem. There are three approaches that have been employed so far (a) Constraint based (b) Score and search (c) Hybrid. The edges of a Bayesian network encode conditional dependency relationships, which constitute the main ingredient of constraint based methods such as the widely used PC algorithm (Spirtes et al., 2000; Kalisch and Bühlmann, 2007). Starting from a complete directed graph, we can use conditional independence tests to recursively decide which edges to keep or discard. These tests start from pairwise comparisons and increase in complexity. Although these tests scale well with increasing dimensions they are prone to local errors and the orders in which they are run (Colombo and Maathuis, 2014). On the other hand, score and search methods rely on some measure of fit of a graphical model to the observed data. These methods then search for the best fitting network that explains the data. Score and search methods are the main focus of our work and we will present some of the challenges in detail in the following sections. Hybrid approaches try to exploit strengths of both the methods for example max-min-hill-climbing method of Tsamardinos et al. (2006), which tries to limit the structural space by learning the skeleton first and then using greedy hill climbing to search for the best fitting model.

We focus on score and search methods, particularly on MCMC methods for sampling graph structures in order to employ Bayesian model averaging to estimate features of interest in the true underlying model. This offers us a major advantage when the amount of data available is modest and the dimensionality is high, where no single best model can be clearly identified. Madigan et al. (1995) presented the first MCMC algorithm, *Structure MCMC*, over graph structures which was later refined by Giudici and Castelo (2003). In order to improve slow mixing and convergence rates of Structure MCMC, Friedman and Koller (2003) suggested building Markov chain over node orderings. Since, space of node orderings, $O(n!)$ is smaller as compared to the super-exponential structure space, it significantly improved upon the convergence and mixing rates at the cost of introducing bias in the sampling. We refer to this as *Order MCMC* in the rest of the document. Grzegorczyk and Husmeier (2008) proposed an edge reversal move to avoid the bias of Order MCMC and simultaneously keeping the convergence rate within reasonable limits. Our approach uses ideas from both Order MCMC of Friedman and Koller (2003) and edge reversal approach of Grzegorczyk and Husmeier (2008) to avoid the bias without compromising the convergence rate. We present our approach, referred to as *Combined DAG Order MCMC* in chapter 4. Kuipers and Moffa (2016) present another novel approach designed on combinatorial structure of DAGs which improves convergence with respect to Structure MCMC while avoiding any kind of bias in sampling. Furthermore, this approach can be combined with edge reversal of Grzegorczyk and Husmeier (2008) to promote convergence. Apart from MCMC techniques, a dynamic programming approach of Koivisto and Sood (2004) can be used for smaller systems. This can also be further extended to propose moves in Structure MCMC (Eaton and Murphy, 2012).

In this chapter, we present some relevant details on the following approaches (i) Structure MCMC (ii) Order MCMC (iv) Partition MCMC, before presenting our approach, Combined DAG Order MCMC, in the next chapter. We use these methods as references to compare the performance of our method on certain aspects in chapter 5.

3.1 Structure MCMC

Structure MCMC was the first algorithm using MCMC methods of sampling graph structures and was proposed by Madigan et al. (1995) and later refined by Giudici and Castelo (2003). It generates a sample of DAGs $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ from the posterior distribution using Metropolis-Hastings sampler in the space of DAGs. Given a DAG, G_i, in the first step a new DAG, G_{i+1} is proposed with the following following proposal probability.

$$Q(G_{i+1}|G_i) = \begin{cases} \dfrac{1}{|\mathcal{N}(G_i)|}, \; G_{i+1} \in \mathcal{N}(G_i) \\ 0, otherwise \end{cases} \tag{3.1}$$

where $\mathcal{N}(G_i)$ denotes the neighborhood of G_i, that is the collection of all the DAGs that can be reached from G_i by (i) Addition (ii) Deletion (iii) Reversal of one single edge of the current graph G_i and $|\mathcal{N}(G_i)|$ is the cardinality of the set. It must be noted that the new graph must be acyclic and thus it must be checked which edge operations can be performed to calculate the neighborhood. In the Metropolis-Hastings algorithm, the proposed move is accepted with the acceptance probability:

$$\alpha(G_{i+1}|G_i) = min\,\{1, R(G_{i+1}|G_i) \tag{3.2}$$

where,

$$\begin{aligned} R(G_{i+1}|G_i) = & \qquad \frac{P(G_{i+1}|D)}{P(G_i|D)} \cdot \frac{Q(G_i|G_{i+1})}{Q(G_{i+1}|G_i)} \\ = & \quad \frac{P(D|G_{i+1})P(G_{i+1})}{P(D|G_i)P(G_i)} \cdot \frac{|\mathcal{N}(G_i)|}{|\mathcal{N}(G_{i+1})|} \end{aligned}$$

If the proposed move is not accepted, then $G_{i+1} = G_i$. Hence, $\{G_i\}$ is a Markov chain in the space of DAGs whose Markov transition kernel, $\mathcal{K}(G_{i+1}|G_i)$ for a move from $G_i \rightarrow G_{i+1}$ is given by

$$\mathcal{K}(G_{i+1}|G_i) = Q(G_{i+1}|G_i) \cdot \alpha(G_{i+1}|G_i)$$

It can be seen that the Markov chain satisfies the condition of detailed balance,

$$\frac{P(\tilde{G}|D)}{P(G|D)} = \frac{\mathcal{K}(\tilde{G}|G)}{\mathcal{K}(G|\tilde{G})}$$

Under ergodicity, it is sufficient condition for the Markov chain $\{G_i\}$ to converge and the posterior distribution $P(G|D)$ is the stationary distribution,

$$P(\tilde{G}|D) = \sum_{G} \mathcal{K}(\tilde{G}|G)P(G|D)$$

In order to make the calculation of neighborhood tractable, especially in high dimensions, we impose an upper limit on the cardinality of parent sets for the nodes. This is referred to as *fan-in*. This reduces computational complexity and improves the convergence of the method. Now, we move onto how to estimate features from the collected samples.

Feature Estimation

Let $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ be the set of collected samples. We can simply use Bayesian model averaging to estimate various features. For the remainder of the thesis we focus only on two specific features (i) Edge estimation, $P(i \to j|D)$ (ii) Parent set estimation, $P(Pa_i = \mathbb{U}|D)$

Edge Estimation

Let use define an edge indicator function, $f(i,j,G)$ that indicates whether or not ∃ an edge from i to j in DAG, G as follows.

$$f(i,j,G) = \begin{cases} 1, \ if \ (i,j) \in E(G) \\ 0, otherwise \end{cases}$$

$E(G)$, refers to the set of edges in DAG, G and the ordered pair (i,j) implies an edge directed from i to j in G. Then, the estimate of probability of the feature given data can be calculated as follows,

$$P(f|D) = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} f(i,j,G)P(G|D) \tag{3.3}$$

Parent Set Estimation

Let us define a parent set indicator function, $f(X_i, G, \mathbb{U})$ that indicates whether the parent set of a node X_i in DAG, G, denoted by $Pa_G(X_i)$ is equal to a given set $\mathbb{U}$ as follows.

$$f(X_i, G, \mathbb{U}) = \begin{cases} 1, \ if \ Pa_G(X_i) = \mathbb{U} \\ 0, \ otherwise \end{cases}$$

Then the estimate of probability of the feature given the observed data is,

$$P(Pa_G(X_i) = \mathbb{U}|D) = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} f(X_i, G, \mathbb{U}) P(G|D)$$

3.2 Order MCMC

It is very clear from the previous section that since Structure MCMC has to walk through entire structural space which is super-exponential, it is slow in mixing of chain and hence, convergence. Additionally, the posterior distribution in structure space can be very jagged with high peaks surrounded by low valleys, which further affects the mixing of Markov chain in structure space. Friedman and Koller (2003) proposed constructing Markov chain in order space, $O(n!)$ instead of structure space. Let us first define a node ordering as any combination of the nodes of a DAG, G such that if $X_i \in Pa_G(X_j)$ then X_i precedes X_j in $\prec$, denoted as $X_i \prec X_j$. For the sake of convenience we use indices of the random variables to denote the nodes in a DAG, G. Therefore, in the remainder of this document, we denote an order as an ordered tuple, $\prec = (k_1, k_2, \dots, k_N)$, where k_i is the index of some node in DAG, G. Friedman and Koller (2003) show that there exists a closed form solution to calculate likelihood of an order given data, as follows

$$\begin{aligned} P(D| \prec) &= \sum_{G \in \mathcal{G}_{k,\prec}} \prod_i score(X_i, Pa_G(X_i)|D) \\ &= \prod_i \sum_{U \in \mathcal{U}_{i,\prec}} score(X_i, U|D) \end{aligned} \tag{3.4}$$

where, $\mathcal{G}_{k,\prec}$ is the set of DAGs with an imposed *fan-in* limit of k that are consistent with the order $\prec$ and $\mathcal{U}_{i,\prec}$ is the possible parents of node X_i defined as, $\mathcal{U}_{i,\prec} :- \{U : U \prec X_i, |U| \leq k\}$.

Order MCMC constructs a Markov chain, $\mathcal{M}$ in the state space of node orderings with the stationary distribution $P(\prec|D)$. In order to ensure that the chain is reversible with the desired stationary distribution by introducing a proposal probability $q(\prec' \mid \prec)$ and, thus, accepting a proposal move with the probability:

$$min\left[1, \frac{P(\prec'|D)q(\prec \mid \prec')}{P(\prec|D)q(\prec' \mid \prec)}\right]$$

Order MCMC introduces two operations for proposal moves: (i) Flip (ii) Cut the deck. Flip operation randomly selects two nodes and flips them to propose a new order. For example, let us assume that k_i and k_j we selected for flip operation. Then we have,

$$(k_1, \ldots, k_i, \ldots, k_j, \ldots, k_N) \rightarrow (k_1, \ldots, k_j, \ldots, k_i, \ldots, k_N)$$

Cut the deck proposes selecting a node randomly and interchanging the two decks of nodes similar to the one used in cards. If k_i is the node selected then we have,

$$(k_1, \ldots, k_i, k_{i+1}, \ldots, k_N) \rightarrow (k_{i+1}, \ldots, k_N, k_1, \ldots, k_i)$$

It can be seen that in both the cases the proposal probabilities are the same as a result, their corresponding terms cancel out in the acceptance probability. Additionally, the chain is reversible, positive recurrent and aperiodic. Hence, we can say that the Markov chain thus constructed has the stationary probability distribution $P(\prec|D)$.

Feature Estimation

Order MCMC uses Metropolis-Hastings sampler to collect samples of node orderings $\prec$, which can then be used to estimate features like presence of an edge in the true underlying model or an estimate of a given set being parent set for a particular node. Since Order MCMC samples orders instead of graph structures, feature estimation calculations are not as simple as in Structure MCMC. In general, if $f(.)$ is any feature, we are interested in

$$P(f| \prec, D) = \frac{P(f, D| \prec)}{P(D| \prec)} \tag{3.5}$$

3.4 shows how to evaluate denominator. The numerator is the sum over all structures that contain the feature and are consistent with the order:

$$P(f|D,\prec) = \sum_{G\in\mathcal{G}_{k,\prec}} f(G)P(G|\prec)P(D|G)$$

Parent Set Estimation

Let us first turn our attention to a simpler problem of estimating that a given parent set, $\mathbb{U}$ is parent of a node X_i, $Pa_G(X_i) = \mathbb{U}$. It can be easily derived using equation 3.5,

$$P(Pa_G(X_i) = \mathbb{U}|D,\prec) = \frac{score(X_i,\mathbb{U}|D)}{\sum_{\mathbb{U}'\in\mathcal{U}_{i,\prec}} score(X_i,\mathbb{U}'|D)} \tag{3.6}$$

Edge Estimation

For calculation of edge features, $j \to i$(say), we apply the same closed form analysis as in the previous section with the only difference that instead of focusing on $\mathcal{U}_{i,\prec}$, we restrict our attention to the subsets of $\mathcal{U}_{i,\prec}$ that contain X_j. Hence,

$$P(j\to i|\prec,D) = \frac{\sum_{\{\mathbb{U}\in\mathcal{U}_{i,\prec}:X_j\in\mathbb{U}\}} score(X_i,\mathbb{U}|D)}{\sum_{\mathbb{U}'\in\mathcal{U}_{i,\prec}} score(X_i,\mathbb{U}'|D)} \tag{3.7}$$

3.3 Partition MCMC

Partition MCMC adheres to the philosophy of Order MCMC but it respects the combinatorial structure of DAGs. It defines Markov chain in the space of node partitions which is essentially a subdivision of node orders in order to avoid over representing certain DAGs. This subdivision does slow the convergence rate as compared to Order MCMC but does not introduce any bias. Additionally, this method can also be combined with the edge reversal technique of Grzegorczyk and Husmeier (2008) to further improve upon the MCMC sampler. Partition MCMC introduces the idea of recursively building DAGs using the concept of outpoints.

Outpoints

Any node that does not have any incoming arcs is called an *outpoint*, also known as *sources*. DAG, because of its acyclic nature, must contain at least one outpoint. If we remove all the edges from all the outpoints of a DAG, we are left with a smaller DAG. We can, thus, recursively build and enumerate DAGs, which also implies that they can be sampled uniformly.

If we combine the outpoints at every stage into m sets each of size k_i, we have, $\sum_{i=1}^{m} k_i = N$ and we have partitioned DAG into $[k_1, k_2, \ldots, k_m]$. By the nature of construction, edges can only flow from a partition to the left into partition to the right. Additionally, there can be no edge within the nodes in the same partition because, otherwise, the node was not an outpoint in the first place. And, conversely, for each node in a partition, there has to be at least one edge from nodes in the adjacent partition because, otherwise, this node was an outpoint for the previous partition. The number of DAGs belonging to a given partition follows from the number of edge possibilities. Let $S_j = \sum_{i=j}^{m} k_i$, then for a partition, $[k_1, \ldots, k_m]$,

$$a_{[k_1,k_2,\ldots,k_m]} = \frac{n!}{k_1! \ldots k_m!} \prod_{j=1}^{m-1} \left(2^{k_{j+1}} - 1\right) \prod_{j=1}^{m-2} 2^{k_j S_{j+2}} \tag{3.8}$$

This method extends the idea of the MCMC scheme proposed by Kuipers and Moffa (2015b) by sampling from the posterior distribution. It does so by assigning a score a_P to a partition P and using the ratio $\frac{a_{P'}}{a_p}$ in the acceptance probability . The combinatorial structure of the DAGs consists of (i) an ordered partition of N, namely $\lambda = [k_1, k_2, \ldots, k_m]$ with $\sum_i k_i = N$ (ii) a permutation π on the node labels (iii) edges connecting the nodes, under certain restrictions. Since the ordering within a partition does not matter, therefore, for a given partition and permutation an ordering can be fixed. We denote π_λ as a single representative for all the equivalent permutations with respect to a partition λ and refer to the ordered pair $(\pi_\lambda, \lambda) = \Lambda$ as *labeled partition.* Under the assumptions, (i) structure modularity (ii) global parameter independence (iii) parameter modularity, we have

$$P(\Lambda|D) = \sum_{\mathcal{G}} P(\Lambda|\mathcal{G}, D)P(\mathcal{G}|D) \equiv \sum_{\mathcal{G}\in\Lambda} P(\mathcal{G}|D) \propto \prod_{i=1}^{n} \sum_{Pa_i \in \Lambda} score(X_i, Pa_G(X_i)|D)$$

Partition MCMC proposes a Markov chain on joint space of partitions and permutations with an acceptance probability,

$$\alpha(\Lambda'|\Lambda) = min\left\{1, \frac{\#nbd(\Lambda)P(\Lambda'|D)}{\#nbd(\Lambda')P(\Lambda|D)}\right.$$

Partition MCMC proposes a basic move comprising of splitting a partition into two or joining two adjacent partitions. While splitting a partition, only the nodes to the left of the newly formed partition need to be re scored. Similarly, during joining of two partitions, only the nodes from the partition element on the left and its adjacent element further left need to be re-scored. Kuipers and Moffa (2016) show that Λ' can be sampled uniformly from the neighborhood of Λ. It can be easily seen that the moves are reversible, aperiodic and positive recurrent and the acceptance probability introduces detailed balance. Therefore, we can use the Markov chain to sample from a stationary distribution $P(\Lambda|D)$. Kuipers and Moffa (2016) also propose additional partition moves like by splitting one partition and then joining one of the splits into another partitions.

In this chapter, we presented some of the previously proposed methods that address the problem of learning Bayesian network structures from data. Specifically, we focused on (i) Structure MCMC (ii) Order MCMC (iii) Partition MCMC as we have used these approaches as a reference point to compare results produced by our method. Being some of the popular approaches in the category of *score and search* methods, they provide a fair ground for comparison and analysis of the performance of our method. We now present our method, *Combined DAG Order MCMC* in the next section.

CHAPTER 4. COMBINED DAG ORDER MCMC

In this chapter, we propose another score and search method which uses both node ordering and DAG structures to build a Markov chain with a target stationary distribution of posterior $P(G|D)$. We refer to it as *Combined DAG Order MCMC*. Our method takes its inspiration from Order MCMC technique of Friedman and Koller (2003) and uses node ordering to heuristically make jumps in the super-exponential structure space and then sample graphs consistent with the order. Unlike Order MCMC, Combined DAG Order MCMC does not use the order modular prior assumption rather a structure modular prior and is therefore free of any bias introduced by Order MCMC. Likewise Structure MCMC, our method samples DAG structures directly from the posterior but makes bigger jumps in the structure space in order to achieve faster rate of convergence as compared to Structure MCMC approach. This is achieved at cost of computation and, thus, our method runs slower in comparison to Order MCMC or Partition MCMC giving a comparable performance to Structure MCMC in regards to computation time. In order to increase efficiency in terms of computation time, our method can be combined with local moves of Structure MCMC. In the following sections, we present our method along with how we can use it to estimate some of the features in the true underlying model in section 4.1 followed by the underlying MCMC methodology detailing the proposal moves in section 4.2. We then present the algorithm used for our experiments in section 4.2 followed by some of the computation tricks employed for making the algorithm computationally tractable in section 4.4. We have devoted another section 4.3 to provide details on how to combine our method with Structure MCMC proposal moves. In the end, we conclude this chapter with an overview of the methodology that sets the ground for experimental results presented in the next chapter.

4.1 Theory

Recall that, a Bayesian network is a DAG G that encodes a joint probability distribution over a set $\mathbb{X} = \{X_1, X_2, \ldots, X_N\}$ of random variables with each variable having a one-to-one mapping with the nodes of DAG G. For convenience, we will always work on the index set $\mathbb{V} = \{1, 2, \ldots, N\}$, where a variable X_i is represented by its index i. We use $Pa_i \subset \mathbb{X}$ to represent the parent set of node represented by X_i. Assume that we are given a data set $D = \{x^1, x^2, \ldots, x^m\}$ with m data points where each x^i is a particular instantiation over the set of variables $\mathbb{X}$. In our approach, we only consider the case of *complete data*, that is all the values are observed and we do not have any missing values.

In Bayesian approach, we compute the posterior probability of a network structure G as

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

We can then estimate any feature of interest defined as a function $f(G)$ defined $\forall G \in \mathbb{S}$ by averaging over all possible networks (1.2 and 2.2.1) as follows.

$$P(f|D) = \sum_{G \in \mathbb{S}} f(G)P(G|D)$$

In this thesis, we mainly focus on the structural features of the true underlying model such as presence of an edge $i \rightarrow j$ that represents a causal relationship between random variables X_i and X_j with X_i being the cause and a probability estimate that a given set $\mathbb{U}$ is the parent set of a node i in the true underlying model.

Our method assumes Structure modular prior,

$$P(G) = \prod_{i=1}^{N} Q_i(Pa_i)$$

where, $Q_i(Pa_i)$ can be any function specifying our prior belief of Pa_i being the parent set of i. There have been many priors proposed such as uniform prior over structures of Heckerman (1996) or by Buntine (1991) that penalizes dense networks. We use the following,

$$Q_i(Pa_i) = \binom{N-1}{|Pa_i|}^{-1}$$

Under the assumptions (i) structure modularity (ii) global parameter independence (iii) parameter modularity, we have

$$P(D|G) = \prod_{i=1}^{N} P(X_i|Pa_i, D) = \prod_{i=1}^{N} score_i(Pa_i|D)$$

where $score_i(Pa_i|D)$ is the local score of node i given its parent set and data. We use *bdeU score* for discrete domains and *bge score* for continuous domains in all our experiments as they satisfy all the mentioned assumptions.

We define an order $\prec$ of the variables as a total order on $\mathbb{V}$ such that if a node i precedes a node j, denoted by $i \prec j$ in the order, then i is present to the left of j in the node ordering specified by $\prec = (k_1, \ldots, i, \ldots, j, \ldots, k_N)$. Let $\mathbb{S}$ define set of all DAG structures over $\mathbb{V}$ and $\mathcal{G}_\prec$ define set of all DAGs consistent with the node ordering $\prec$. Given a distribution $P(G)$ over DAG space $\mathbb{S}$, we can consider an order $\prec$ as the event $\mathcal{G}_\prec \subset \mathbb{S}$ and compute $P(\prec)$. It must be noted that for different orders $\mathcal{G}_\prec$ overlaps, therefore, orders are not mutually exclusive events. As a consequence, Order MCMC is biased.

$\forall\ Pa_i \subset \mathbb{V}$, we define

$$\beta_i(Pa_i) \equiv Q_i(Pa_i) score_i(Pa_i|D) \tag{4.1}$$

We then define the following function $\forall i \in \mathbb{V}$, $\forall\ S \subseteq \mathbb{V} - \{i\}$,

$$\alpha_i(S) \equiv \sum_{Pa_i \subseteq S} \beta_i(Pa_i) \tag{4.2}$$

Considering $\prec$ to be an event of selecting a graph consistent with the order $\prec$ from the structure space. We have,

$$\begin{aligned} P(\prec, D) = & \sum_{G \in \mathcal{G}_\prec} P(D|G)P(G) \\ = & \sum_{G \in \mathcal{G}_\prec} \prod_{i=1}^{N} \beta_i(Pa_i) \\ \Longrightarrow P(\prec, D) = & \prod_{i=1}^{N} \alpha_i(U_i^\prec) \end{aligned}$$

where $U_i^{\prec}$ is the set of variables preceding i in order $\prec$. The posterior of a particular choice of parent set $Pa_i \in U_i^{\prec}$ for a node i is given by

$$P(Pa_i | \prec, D) = \frac{\beta_i(Pa_i)}{\alpha_i(U_i^{\prec})} \tag{4.3}$$

We define another function,

$$\alpha_i^*(S|j) = \alpha_i(S) - \alpha_i(S - \{j\}) \tag{4.4}$$

For any $j \in U_i^{\prec}$, posterior that X_j is parent of X_i(or the posterior of an edge feature $j \to i$) is given

$$P(j \to i| \prec, D) = \frac{\sum_{Pa_i \subseteq U_i^{\prec}, j \in Pa_i} \beta_i(Pa_i)}{\alpha_i(U_i^{\prec})} = \frac{\alpha_i^*(U_i^{\prec}|j)}{\alpha_i(U_i^{\prec})} \tag{4.5}$$

For any $j \in U_i^{\prec}$, the posterior of a given set $Pa_i = \mathbb{U} \subseteq U_i^{\prec}$ for a node i that must contain j is given by

$$P(Pa_i = \mathbb{U}| \prec, D, j \to i) = \frac{P(Pa_i| \prec, D)}{P(j \to i| \prec, D)} = \frac{\beta_i(Pa_i)}{\alpha_i^*(U_i^{\prec}|j)} \tag{4.6}$$

In this section, we have presented the theory behind feature estimation in terms of node ordering in a graph using a structure modular prior. Next, we present construction of Markov chain in structure space using node orderings to make proposal moves. Equations 4.3 and 4.6 play an important role in sampling of a DAG given an order as shown in the following section.

4.2 MCMC Methodology

In chapter 2, we have seen that MCMC methods sample from a target stationary distribution by constructing a Markov Chain. Under certain assumptions that chain eventually converges to the stationary distribution from which samples can be obtained. Combined DAG Order MCMC attempts at sampling DAG structures from the posterior $P(G|D)$ by constructing a Markov chain and using both the node ordering as well as the DAG structure to make proposal moves in the chain. It uses a two step approach to propose a new move from a given DAG to another DAG. Starting with a random order $\prec$ and a DAG consistent with the order $\prec$ denoted by $G_{\prec}$, first step involves choosing an edge $j \to i$ at random to identify two nodes in the node ordering and flipping them to propose a new order $\prec'$. In the second step, we sample a new

DAG $G_{\prec'}$ consistent with the new order. To sample a DAG from a given order, we sample parents for each node as per node ordering using the available data that is high scoring parents are sampled with a higher probability. It must be noted that in order ensure the convergence of chain, we need every move to be reversible. Thus, we ensure the presence of an edge $j \leftarrow i$ in the resulting DAG.

We use the following proposal moves (i) Flip and sample modified(*SimpleFlip*) (ii) Flip and sample all, (*FlipSampleAll*). At the begining of each step we can choose one move at random and then sample a DAG accordingly. If *SimpleFlip* is selected, we pick an edge at random, say $j \rightarrow i$ in a graph $G_{\prec}$ consistent with an order $\prec = (k_1, \ldots, j, \ldots, i, \ldots, k_N)$ and then flip i and j to obtain a new order $\prec' = (k_1, \ldots, i, \ldots, j, \ldots, k_N)$. This results in change of set of potential parents, we refer to as *potential parent set* for the nodes $\mathbb{K} = \{k \mid j \ \preceq k \preceq i \ \in \prec'\}$. So we sample a new parent set for each node $i \in \mathbb{K}$ from its corresponding potential parent set as per the new order, $\prec'$. From the nature of construction, sampling parent set for each node results in a DAG and therefore no cycle check is required. Sampling is done using equation 4.3 and ensuring $\exists j \leftarrow i$ so that the move is reversible. *FlipSampleAll* is same as *SimpleFlip* with the variation that instead of sampling parents only for the affected nodes we sample parents for all the nodes. At the cost of adding more computations it gives us an advantage of better mixing of chain as compared to *SimpleFlip*. So this move can be carried out with a lower probability as compared to *SimpleFlip*.

Now, let us consider the proposal moves from the perspective of meeting the conditions of convergence for a Markov chain. We intend to construct a Markov chain with a target stationary distribution $P(G|D)$ using the proposed moves. From the nature of the sampling of a new DAG, it is clear that the chain is *aperiodic* and *positive recurrent*, thereby *ergodic*. We stress on ensuring that if $j \rightarrow i$ was the selected edge then the resulting graph must have an edge $j \leftarrow i$. This ensures that the chain is *irreducible* that is there are no islands being formed and all the subsets are connected. In the later sections, we present the our acceptance probability term that we use to ensure detailed balance in our Metropolis-Hastings sampler

(2.2.2). This ensures *reversibility* which implies *stationarity.* Therefore, by using appropriate acceptance probability we can ensure that the Markov chain thus constructed would reach its target stationary distribution $P(G|D)$ after some burn-in iterations. Before presenting the calculations involved, let us first present the proposed algorithm as follows.

Algorithm

We selected the moves *SimpleFlip* and *FlipSampleAll* with probabilities $\{0.70, 0.30\}$ respectively.

```
I. Start with a random order $\prec$ and a DAG, $G_{\prec}$, conforming to the order.

II. Until, samples are collected, do

    a. Randomly pick an edge, $i \rightarrow j$ in $G_{\prec}$.

    b. Flip $i$ and $j$ in $\prec = \{k_1, \ldots, i, \ldots, j, \ldots, k_N\}$ to get a new order $\prec' = \{k_1, \ldots, j, \ldots, i, \ldots, k_N\}$.

    c. Pick $proposalMove$ from $\{\texttt{SimpleFlip, FlipSampleAll}\}$ with probability $\{0.70, 0.30\}$

    d. If $proposalMove == \texttt{FlipSampleAll}$, then

        i. $startIndex \leftarrow 1$

        ii. $endIndex \leftarrow n$

       else,

        i. $startIndex \leftarrow j$

        ii. $endIndex \leftarrow i$

    e. Define, $G_{\prec'} \leftarrow G_{\prec}$

    f. For node, $k \in \{startIndex, \ldots, endIndex\} \subseteq \prec'$,do

        i. Orphan node $k$ in $G_{\prec'}$

        ii. If, $k = i$, then, sample parents of $i$ ensuring an edge from $j \rightarrow i$ using equation
            4.6

        iii. else, Sample parents of, $k$ according to $P(Pa_k | \prec', D)$, as given in equation 4.3.
```

Proposal Probability

We need to study the proposal probabilities for all the moves. The proposal move consists of two steps, first proposes the order and the second a graph. First of all, we pick a random edge in $G_{\prec}$, to propose the new order $\prec'$. If $E_{G_{\prec}}$ denotes the set of edges in $G_{\prec}$, then

$$P(\prec |G_{\prec}, \prec) = \frac{1}{|E_{G_{\prec}}|} \tag{4.7}$$

Then, we sample parents of all the affected nodes, that is the nodes whose potential parent sets have changed owing to the flip operation. This depends upon the position of i and j in the new order $\prec'$. Also, it may be noted that for the node $i \in \prec' = \{1, \ldots, j, \ldots, i, \ldots, n\}$, j is already a parent of i, hence, the probability calculation for i is different.

$$P(G_{\prec'}|j \to i, \prec', D) = P(Pa_i| \prec', j \to i, D) \cdot \prod_{k=j}^{i-1} P(Pa_k| \prec', D) \tag{4.8}$$

Equations 4.3 and 4.5 talk about the details of evaluating 4.8

We denote the proposal probability of going from $G_{\prec} \to G_{\prec'}$ by $Q(G_{\prec'}|G_{\prec})$, as follows:

$$\begin{aligned} Q(G_{\prec'}|G_{\prec}) &= P(\prec' |G_{\prec}, \prec) \cdot P(G_{\prec'}|j \to i, \prec', D) \\ &= \frac{1}{|E_{G_{\prec}}|} \cdot P(Pa_i| \prec', j \to i, D) \cdot \prod_{k=j}^{i-1} P(Pa_k| \prec', D) \end{aligned}$$

$$\Longrightarrow Q(G_{\prec'}|G_{\prec}) = \frac{1}{|E_{G_{\prec}}|} \cdot \frac{\beta_i(Pa_i)}{\alpha_i^*(U_i^{\prec'}|j)} \cdot \prod_{k=j}^{i-1} \frac{\beta_k(Pa_k)}{\alpha_k(U_k^{\prec'})} \tag{4.9}$$

Similarly, for *FlipSampleAll*,

$$\Longrightarrow Q(G_{\prec'}|G_{\prec}) = \frac{1}{|E_{G_{\prec}}|} \cdot \frac{\beta_i(Pa_i)}{\alpha_i^*(U_i^{\prec'}|j)} \cdot \prod_{\substack{k=1 \\ k \neq i}}^{n} \frac{\beta_k(Pa_k)}{\alpha_k(U_k^{\prec'})} \tag{4.10}$$

From here on, for simplicity reasons, let us use the terms sI for *startIndex* and eI for *endIndex* to refer to the range of indices of the nodes whose parents need to be sampled. We reserve i and j to refer to the nodes which were selected to be flipped. Equations 4.9 and 4.10, can therefore be simplified to

$$Q(G_{\prec'}|G_{\prec}) = \frac{1}{|E_{G_{\prec}}|} \cdot \left(\prod_{k=1}^{n} \beta_k(Pa_k) \right) \cdot \left(\prod_{\substack{k=sI \\ k \neq i}}^{eI} \frac{1}{\alpha_k(U_k^{\prec'})} \right) \cdot \frac{1}{\alpha_i^*(U_i^{\prec})} \tag{4.11}$$

For simplicity, we use the following notation

- G', G for $G_{\prec'}, G_{\prec}$ respectively.
- Pa'_k, Pa_k to denote parent set of node k in G', G respectively.
- E', E to denote number of edges in G', G respectively.

Before, moving onto the section on acceptance probability, let us consider the term $\dfrac{P(G'|D)}{Q(G'|G)}$. We have

$$P(G|D) \equiv \prod_{k=1}^{n} \beta_k(Pa_k)$$

This simplifies the expression,

$$\frac{P(G'|D)}{Q(G'|G)} = \frac{1}{E} \cdot \left(\prod_{\substack{k=sI \\ k \neq i}}^{eI} \alpha_k(U_k^{\prec'}) \right) \cdot \alpha_i^*(U_i^{\prec'}|j) \tag{4.12}$$

Acceptance Probability

In order to avoid chain swaying either ways, we introduce a balancing factor called *acceptance probability*, which adds the necessary *detailed balanced*, for the chain to converge. After every proposal, we calculate the acceptance probability value, according to which the move is accepted or rejected. It is defined as,

$$A(G'|G) = \min\left\{1, R(G'|G)\right\} \tag{4.13}$$

where,

$$R(G'|G) = \frac{P(G'|D) \cdot Q(G|G')}{P(G|D) \cdot Q(G'|G)}$$

Using equation 4.12, it reduces to

$$R(G'|G) \quad = \quad \frac{E'}{E} \cdot \left(\prod_{\substack{k=sI \\ k \neq i \\ k \neq j}}^{eI} \frac{\alpha_k(U_k^{\prec'})}{\alpha_k(U_k^{\prec})} \right) \cdot \frac{\alpha_j(U_j^{\prec'})}{\alpha_i(U_i^{\prec})} \cdot \frac{\alpha_i^*(U_i^{\prec'}|j)}{\alpha_j^*(U_j^{\prec}|i)} \tag{4.14}$$

Equation 4.14 adds a lot of overhead to our calculations which increases the running time our method. Therefore, we can combine our proposal moves with some of the local moves of Structure MCMC in order to speed up the rate of convergence owing to the bigger jumps proposed by Combined DAG Order MCMC and reduce the time complexity of the algorithm by making use of some of the non-rigorous local moves. The following section talks in detail about this approach.

4.3 Combining with Structure MCMC

We have seen that even the simplest of moves proposed by Combined DAG Order MCMC are computationally expensive. Hence, we propose mixing it with local proposal moves of Structure MCMC. Thus, our method has the ability of make bigger jumps in the structure space and achieve speedy mixing of the chain along with the option of proposing simple local moves of Structure MCMC to reduce the overall computation time. In order to do so, we can select amongst the Structure MCMC moves and Combined DAG Order MCMC moves with some probability mix, say $60 - 40$. Furthermore, we can select amongst *SimpleFlip* and *FlipSampleAll* with a certain probabilities, say $70 - 30$. It must be noted that, Combined DAG Order MCMC considers the node orderings of DAGs as well whereas Structure MCMC has no consideration for that. This introduces a disjoint between the Structure MCMC and our method's proposal moves. To overcome this problem, we propose adding an additional step to the proposal process which includes selecting an order uniformly from all the orders the DAG is consistent with. We take inspiration from Partition MCMC of Kuipers and Moffa (2016) and use the concept of outpoints randomly sample an order a DAG is consistent with. Then we can use our proposal process to propose a move in the chain. Following section illustrates this process.

Uniformly sampling an order

In this section, we consider the problem of uniformly sampling an order a DAG is consistent with. We use the concept of *outpoints* (Kuipers and Moffa, 2016) to sample such an order. Outpoints are defined as nodes that have no incoming edges. By the nature of construction, in a DAG there must exist at least one outpoint. Starting with a DAG, we collect all the outpoints in a set called *partition* and remove all the outpoints and edges emanating out of the outpoints.

This creates another set of outpoints in the subsequent DAG. We follow this process recursively until we are left with no outpoints. Then we uniformly sample nodes without repetition from each partitions and concatenate each sampling for partitions in the same order the partitions were obtained. This results in an order DAG is consistent with. Figure 4.1 illustrates this process. The DAG shown in the example can be broken into 3 partitions (i) at the first stage, nodes 1 and 2 are outpoints. Therefore, collected in *Partition-1* (ii) removing them from the DAG, leaves nodes 3 and 5 as outpoints and collected in *Partition-2* (iii) eventually, we are left with only node 4, which is collected into *Partition-3*. Then we sample nodes uniformly from each partition without repetition and combined them in the same order as partitions were obtained to get an order that the DAG is consistent with. This introduces another term in the calculation of acceptance probability because of the additional sampling step for order. The total number of partitions possible for a DAG can be easily obtained once the partitions have been formed. Using combinatorics, it can be seen that if a DAG, G, is divided into K partitions where size of i^{th} partition is denoted by k_i, the total number of orders is $\mathcal{N}_{\prec}(G) = \prod_{i=1}^{K} k_i!$. If we denote $R'(G'|G)$ as the new acceptance probability then

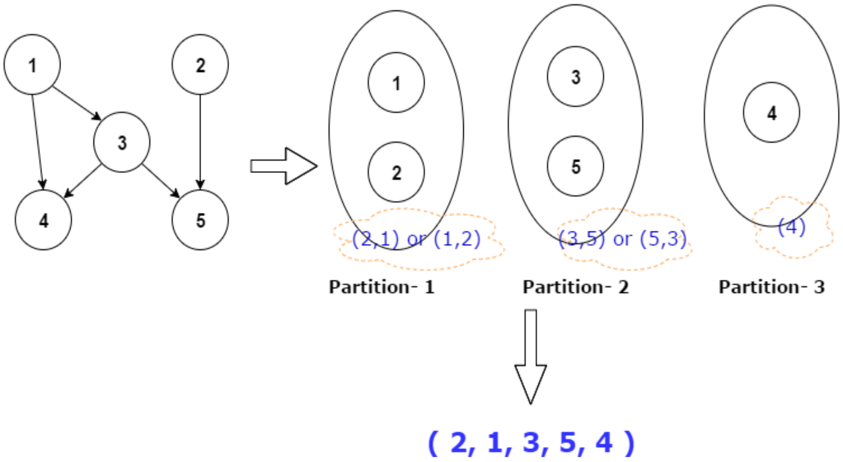


Figure 4.1: Sampling Order given a DAG

$$R'(G'|G) = \frac{\mathcal{N}_{\prec}(G')}{\mathcal{N}_{\prec}(G)} R(G'|G)$$

where $R(G'|G)$ is defined in equation 4.14.

Thus, we have a 3-step process to propose a move whenever Combined DAG Order MCMC move is selected. This enables the chain to make bigger jumps and at the same time reduce the time complexity of the algorithm by using much simpler local moves. In order to tackle some of the issues with time complexity, we used some computational tricks which we present in the next section.

4.4 Computational Tricks

The calculation of acceptance probability is a computation intensive step and slows down this method in comparison to other methods. However, there are a couple of tricks that can be employed in order to save some of the computation steps. Following the footsteps of earlier approaches to improve computation, we define an upper limit to the number of parents a node can have. For our experiments, we set this limit to 4. This reduces the number of families for a node. Furthermore, we pre-compute the scores of every possible family of each node and then use the structural modularity assumption to calculate the score of DAG, as follows

$$P(G|D) = \prod_{i=1}^{n} P(Pa_i|D)$$

As the scores, can be very small, we store logarithms of the values and operate in logarithmic space for most of our calculations. We do not sample the entire graph as per new order, but only the affected nodes to save some computation costs. To further speed up calculations, all the values of $\alpha(S)$ can also be stored in a mapping $sort(S) \rightarrow value$. Additionally, since we need to resample parents of some of the nodes, which require evaluation of possible families for each set of node. Because of the nature of order $\prec$ of nodes and the rules it imposes on the family sets for the nodes, we can construct the families recursively using the family set for the node in the previous set. This approach in conjunction with the map approach can help save a lot of duplicate calculations. Special attention needs to be devoted to the $\alpha(S)$ calculations, as the values being very very small, we need to add them in *logarithmic space*. Otherwise, only one dominant value might get picked always for a particular set and it may affect proper mixing of chain.

In this chapter, we have presented our approach of Bayesian structure learning from data using both node ordering and DAG structures in order to sample DAGs that can be used to estimate some of the features of the true underlying model for a particular domain. Our method samples DAGs directly from the posterior, $P(G|D)$. We propose construction of a Markov chain in structure space with posterior as the target stationary distribution. Proposal is a two step process and the step involving sampling of parents is computationally expensive. There are a couple of tricks we proposed in the chapter in order to reduce this time complexity. Additionally, we have shown how our method can be combined with Structure MCMC in an attempt to reduce the overall time of the algorithm by making use of much simpler local moves. We have conducted experiments using both the approaches (i) Combined DAG Order MCMC (ii) Combined DAG Order MCMC in conjunction with Structure MCMC, we call *New Structure MCMC*. The following section presents the experimental results gathered along with their analysis.

CHAPTER 5. EXPERIMENTAL RESULTS

In this chapter, we presents experiments that were conducted to analyze the effectiveness of our approach. In section 5.1 we present our experimental setup detailing the methods we chose to compare our method against and the specific parameters we used for our underlying Metropolis-Hastings sampler. We present the various criterion on which we evaluated our method in this section. It also specifies the scoring methods used in our technique and the data sets we used for the comparisons. Section 5.2 presents the experimental results which is concluded by our analysis of the results on how our method compares against the state of the art approaches that it was compared against.

5.1 Experimental Setup

As discussed in chapter 3, there are a lot of approaches that have been proposed to tackle the problem of Bayesian structure learning using data. Out of all, Order MCMC and Partition MCMC are the ones that provide an efficient approximation of the solution. Thus, they formed natural choices to analyze the effectiveness of our approach. We choose the following methods for comparisons: (i) Structure MCMC (ii) Order MCMC (iii) Partition MCMC. Order MCMC and Partition MCMC are two state of the art methods which have a very good convergence rate and are perfect candidates for benchmarking the effectiveness of Combined DAG Order MCMC. Structure MCMC was chosen in order to use it as a reference point for all the three methods. Additionally, we compared our results with Partition MCMC using edge reversal move as well. Since, Partition MCMC along with edge reversal move showed almost similar behavior as Partition MCMC for the data sets, therefore, for the sake of simplicity we do not present results of Partition MCMC with edge reversal. We ran experiments for both Combined

DAG Order MCMC as well as its combination with Structure MCMC, which we refer to as *New Structure MCMC* in the experimental results. For the implementation we used github repository of Kuipers and Moffa (2015a) as they already have a very efficient implementation of the methods required for our comparison. We now present details of the data sets used in our results in the following section, which is followed by details on the criteria used for evaluation.

5.1.1 Data Sets

We used the following data sets in order to gather experimental data for analysis: (i) Synthetically generated data set of Kuipers and Moffa (2015a), we refer to as *data5Nodes.* (ii) Asia data set (iii) Sachs data set (iv) Boston Housing data set. Most of the data sets were obtained from *bnlearn* package of R (Scutari, 2010). We used a balance of synthetic and actual data sets for our comparisons. As it is well known that synthetic data sets provide a more controlled environment to test your methods and at the same time we also wanted to evaluate our method against actual data sets. For some of the data sets the true underlying model is known which served as another reference point to evaluate the sampling results. *data5nodes* is an in-house generated data set. It consists of 5 random variables with a continuous domain sampled from a normal distribution with some random noise added. Additionally, it also adds correlation amongst some of the variables. This is same model as used by Kuipers and Moffa (2016). Asia is another synthetic data set from Lauritzen and Spiegelhalter (1988) about lung diseases and visits to Asia. It consists of 8 random variables with discrete domain. Bnlearn package provides this data set along with the estimated underlying model which we used to compare our sampling results. Sachs is another dataset available in bnlearn package that consists of 11 random variables with discrete domain (Sachs et al., 2005). Finally, we ran our results on Boston housing data available at Lichman (2013). This regression dataset concerns housing values in suburbs of Boston, the problem is to predict the median house price given a number of demographic features. It consists of 14 nodes. For the data sets with discrete variables, we used *bdeU* score and for continuous domains we used *bge* scores. All the scores were calculated using *bnlearn* package of Scutari (2010). We now present the evaluation criteria used in our experimental results in the following section.

5.1.2 Evaluation Criteria

We evaluated our method on the following criteria (i) Sampling results (ii) Rate of convergence (iii) Robustness (iv) Feature estimation. Firstly, we evaluated our sampling results from perspective of distribution of the scores of the sampled DAGs and compared with other state of the art methods. We do not want our method to sample sub-optimally scoring DAGs. We also want some variation in the samples so as not to have the chain stuck at single point in the structure space. Variation would vary with respect to the availability of the data and the domain, therefore, we ran a comparison against other methods using the same parameters so as to have an unbiased platform for comparisons. Secondly, we analyzed our method to see how quickly our method reaches its target stationary distribution. Given the super-exponential structure space, rate of convergence is a very important criteria and one of the major motivating factors to study this problem. Thirdly, we analyzed our method from robustness perspective, that is given a different starting point how consistent is the behavior. The reason for this criteria was basically to check if the chain gets stuck in some local optima regions or not. Finally, we wanted to check the performance of our method from the features estimation perspective, which is the main goal of studying the problem. For feature comparison, we used two different setups. In one setup, we compared all the edge features estimated by (i) Combined DAG Order MCMC (ii) New Structure MCMC against true value using Bayesian Model Averaging approach. In order to do so, we used a synthetically generated data set of 5 nodes and using a maximum parent limit of 4, we computed true values of all the features as well as estimates obtained by our methods and compared them. Since, exhaustive enumeration is only possible for smaller domains. Therefore, we also compared estimates from our method with probability estimates obtained via (i) Structure MCMC (ii) Partition MCMC. We were not interested in comparing our results against the ones obtained by Order MCMC due to the bias introduced by the methods. First, we present sampling results.

5.2 Results

5.2.1 Sampling Results

We are interested in the quality of the samples our method provides because our end goal of feature estimation depends solely on the sampling results. Therefore, first and foremost we evaluate the scores of the sampled graphs along with the variability of the sample. This gives us an idea as to how good our samples are. We expect high scoring samples and some variability in the collected samples. The reason for expected variability is that we do not want our method to be stuck in a region of local optima and sample out the same DAG all the time. To do so, we plotted the a density curve of all the scores of sampled DAGs for a particular run. For the sake of easy comparison, we plotted the curves for all the methods on the same plot using a color code. It must be noted that Order MCMC is biased so we are not very interested in the samples obtained from Order MCMC. First of all we present the results on our synthetic data set *data5Nodes*. We collected 1000 samples with a burn-in of 200 iterations and thinning of 10 iterations. Figure 5.1 shows the results. It can be clearly seen that all the methods

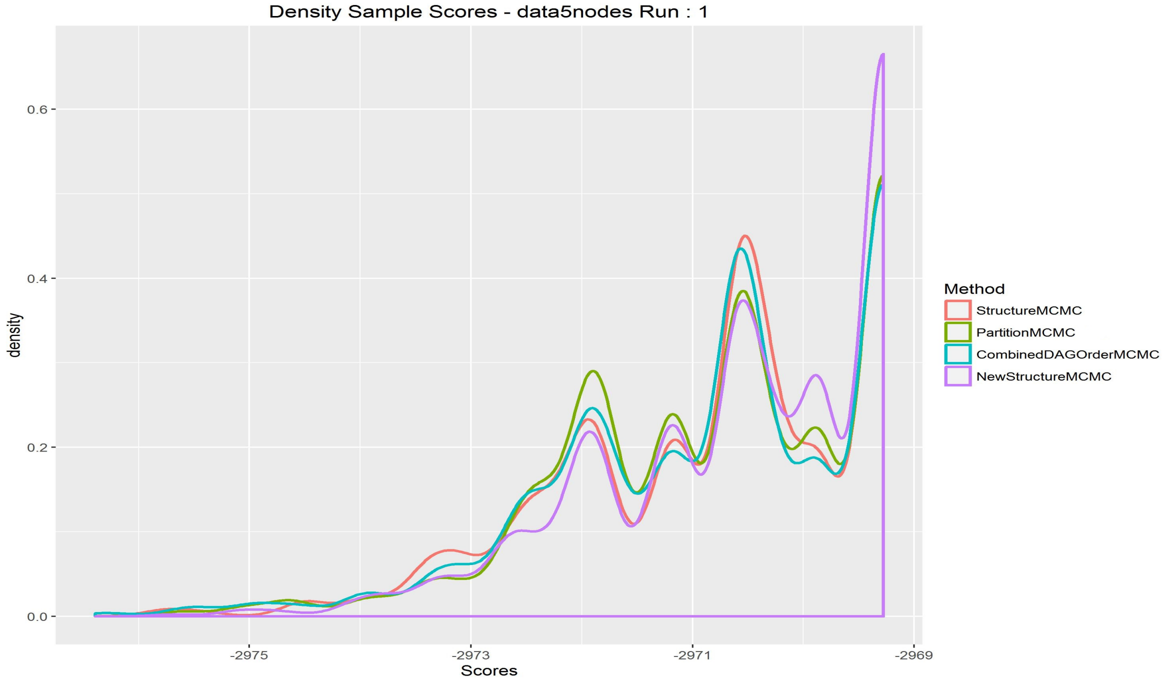


Figure 5.1: Sampling results - data5Nodes

give out similar results. The convergence to the stationary distribution can be seen around the increase in the density of DAGs around a particular score. Next we present the sampling results on the remaining data sets in the order of increasing complexity of the model. For *asia* and *sachs* data set we collected 1000 samples with a burn in of 200 iterations and a thinning of 10 iterations shown in figures 5.2 and 5.3 respectively. For *Boston Housing* data set we used a burn in of 1000 iterations to collect 1000 samples with thinning of 10 iterations as shown in figure 5.4. It can be observed for slow converging methods Structure MCMC and its variant *New Structure MCMC*, as the complexity increases certain peaks come up at a different score from the majority of the collected samples which shows the slow convergence of those methods as compared to others. To conclude, we can say that the sampling results for both Combined DAG Order MCMC and New Structure MCMC are similar to the results given by all the other novel approaches. Next we present rate of convergence results.

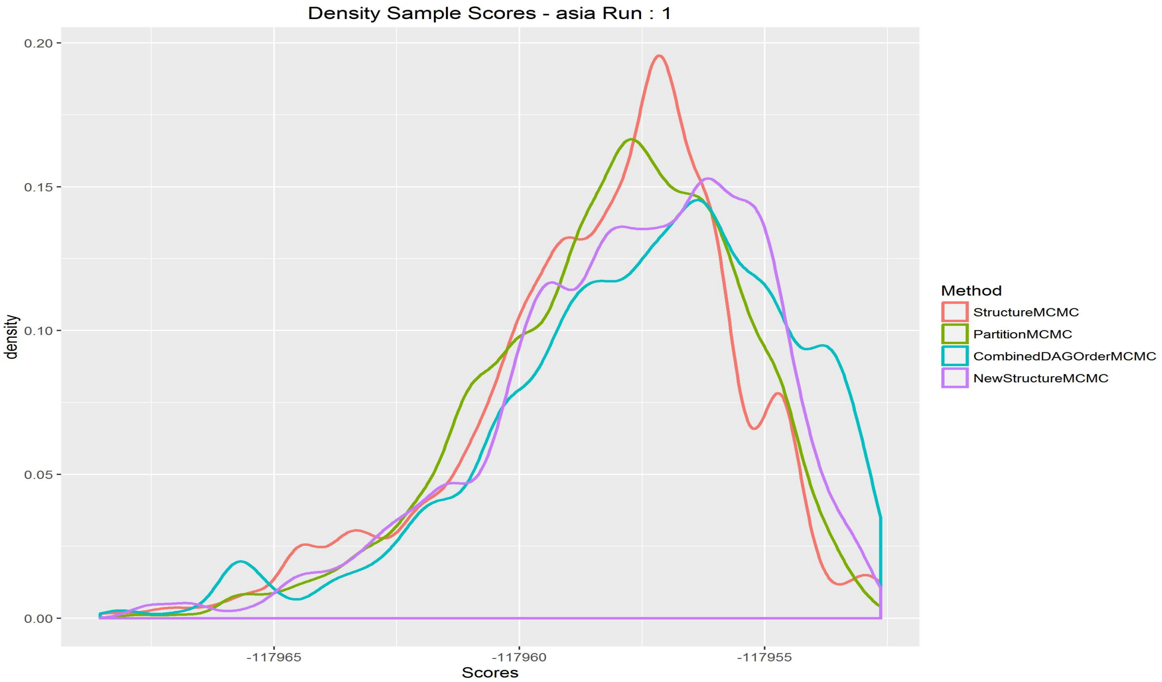


Figure 5.2: Sampling results - asia

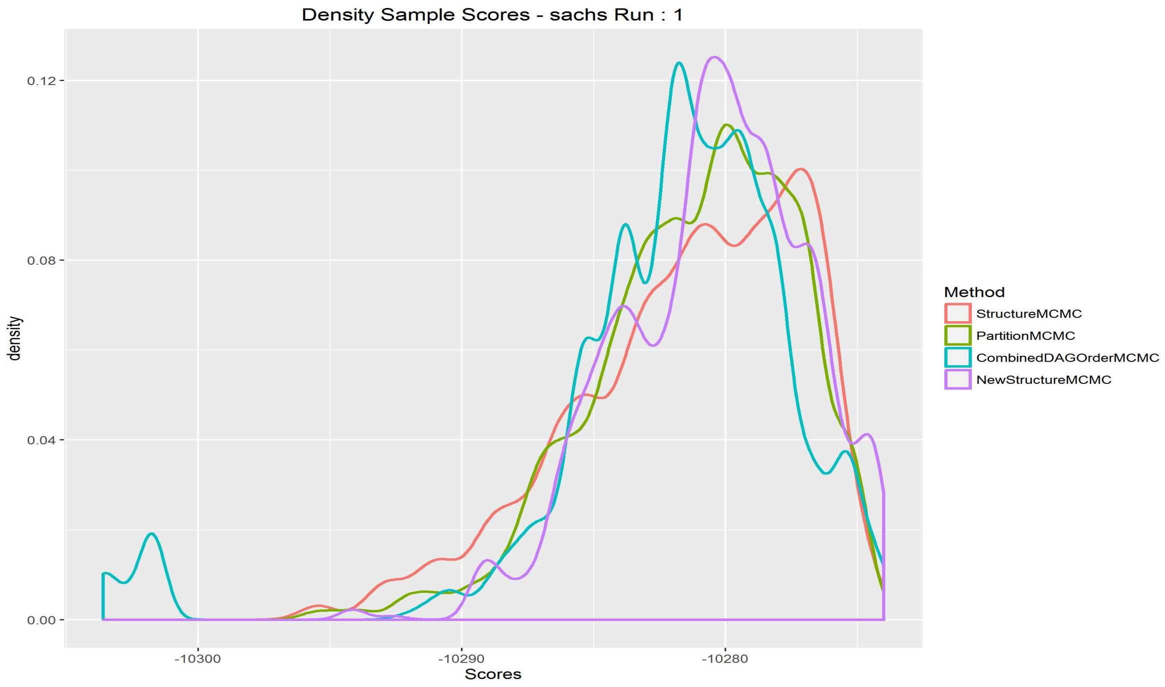


Figure 5.3: Sampling results - sachs

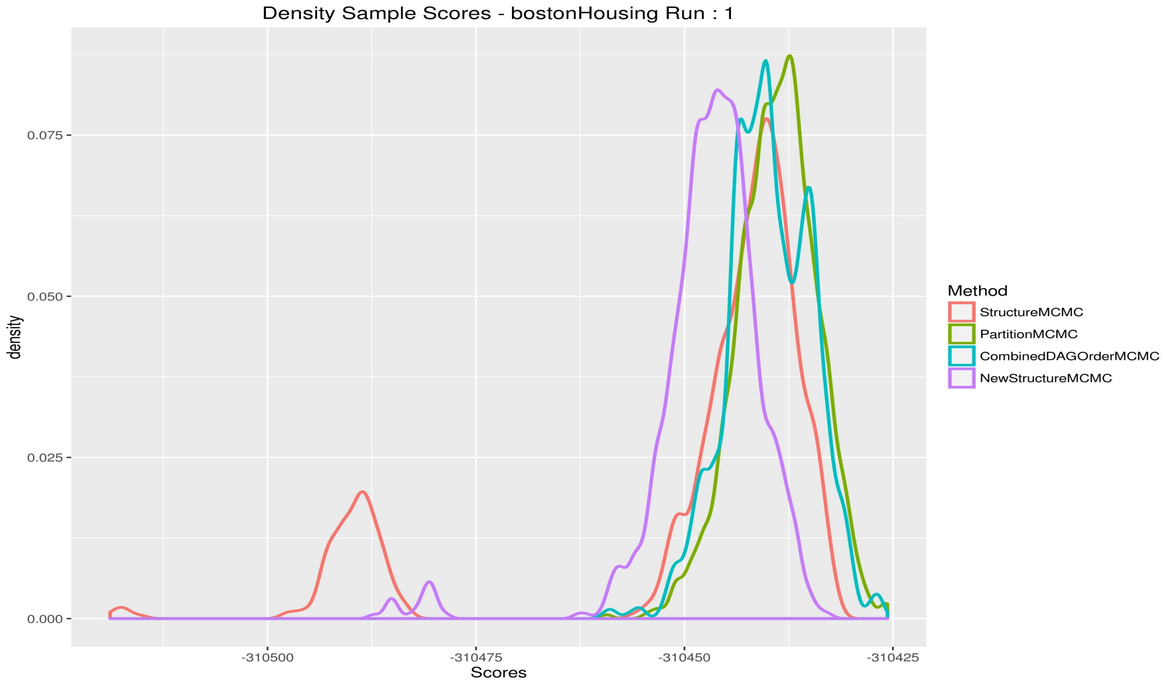


Figure 5.4: Sampling results - Boston Housing

5.2.2 Rate of Convergence

By analyzing the rate of convergence of our algorithm, we want to analyze how quickly the chain attains its target stationary distribution. Because of slow mixing of Markov Chain, MCMC methods are slow to converge. Hence, rate of convergence is an important metric to evaluate the effectiveness of our method. It can be effectively analyzed by plotting the scores of all the proposed moves along the chain. Therefore, we have plotted the posterior value(scores) at each state of Markov Chain on y-axis against the iterations on x-axis. Additionally, we also plotted the burn-in iterations and the score of true DAG(if known) as dashed lines parallel to y-axis and x-axis respectively. For an effective and easy comparison, we plotted the scores for all the methods on the same plot. First, we compare the results on *data5nodes* in figure 5.5. We collected 1000 samples with a burn in of 200 iterations and thinning of 10 iterations. Being a simple model, it can be observed that all the methods converge within burn in iterations.

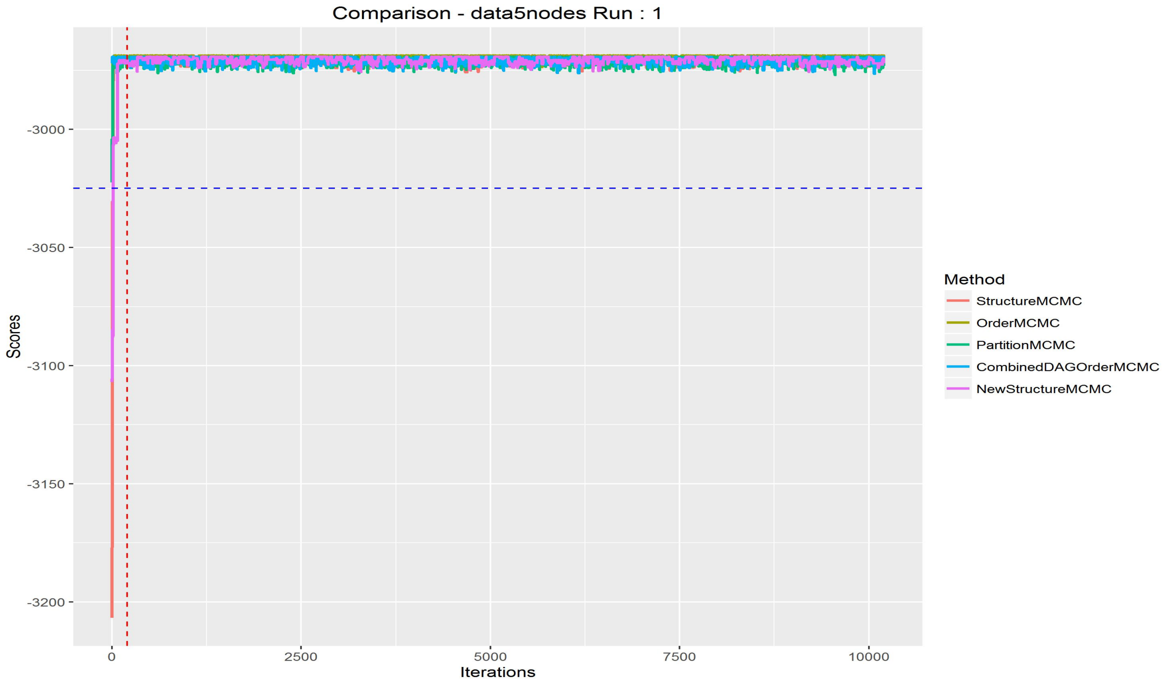


Figure 5.5: Rate of Convergence - data5Nodes

Next we present convergence plots for *asia*, *sachs* and *Boston Housing* data sets in the figures 5.6, 5.7 and 5.8 respectively. For *asia* and *sachs* we used burn in of 200 and thinning of 10 to collect 1000 samples. Boston Housing data set being a complex as compared to others, we set the burn in to 1000 keeping rest of the parameters same. As the complexity increase, Structure MCMC and New Structure MCMC can be seen comparatively slow in converging. This is clearly evident in case of Boston Housing data set, where they take a lot of time to converge even with a burn in of 1000 iterations. New Structure MCMC is faster as compared to Structure MCMC owing to bigger jumps introduced by Combined DAG Order move. It can be concluded that both the methods have provided expected results and all the convergences around the known good DAG score further strengthen our belief in the quality of results. Next we analyze the robustness of the methods.

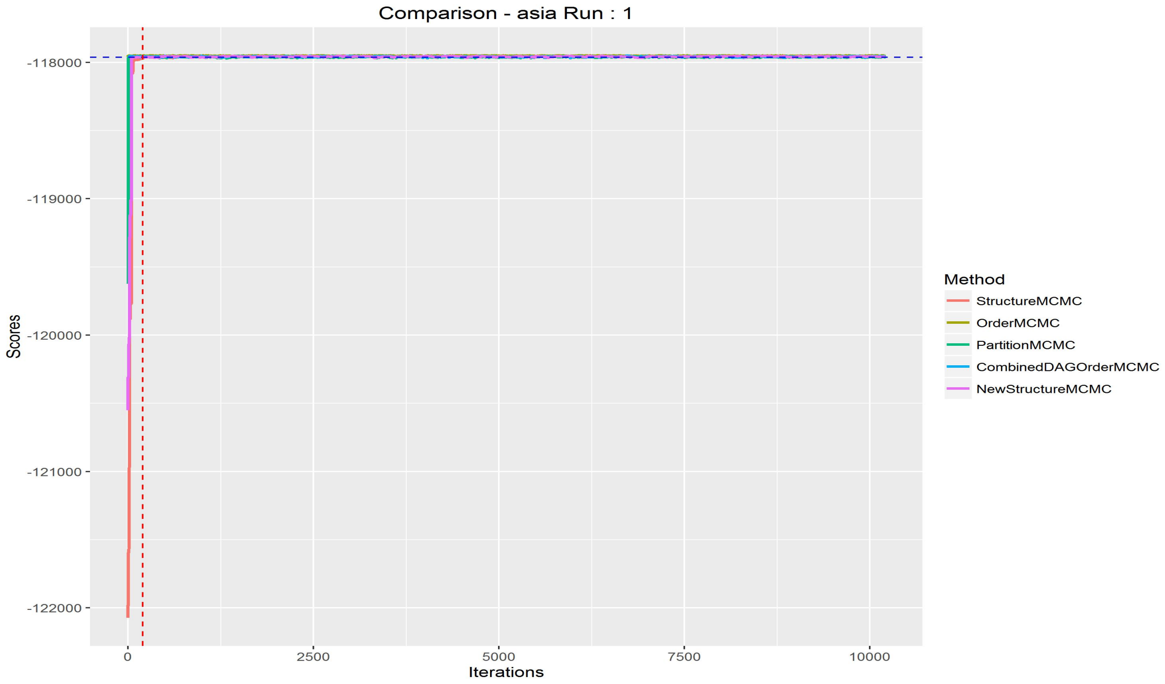


Figure 5.6: Rate of Convergence - asia

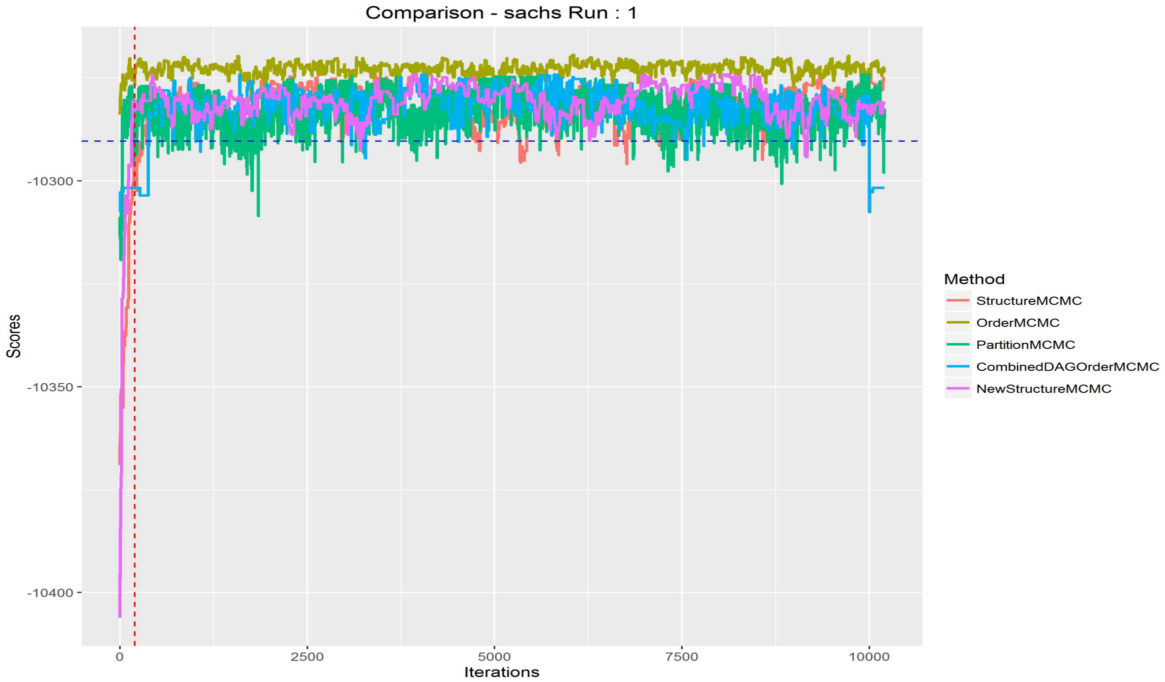


Figure 5.7: Rate of Convergence - sachs

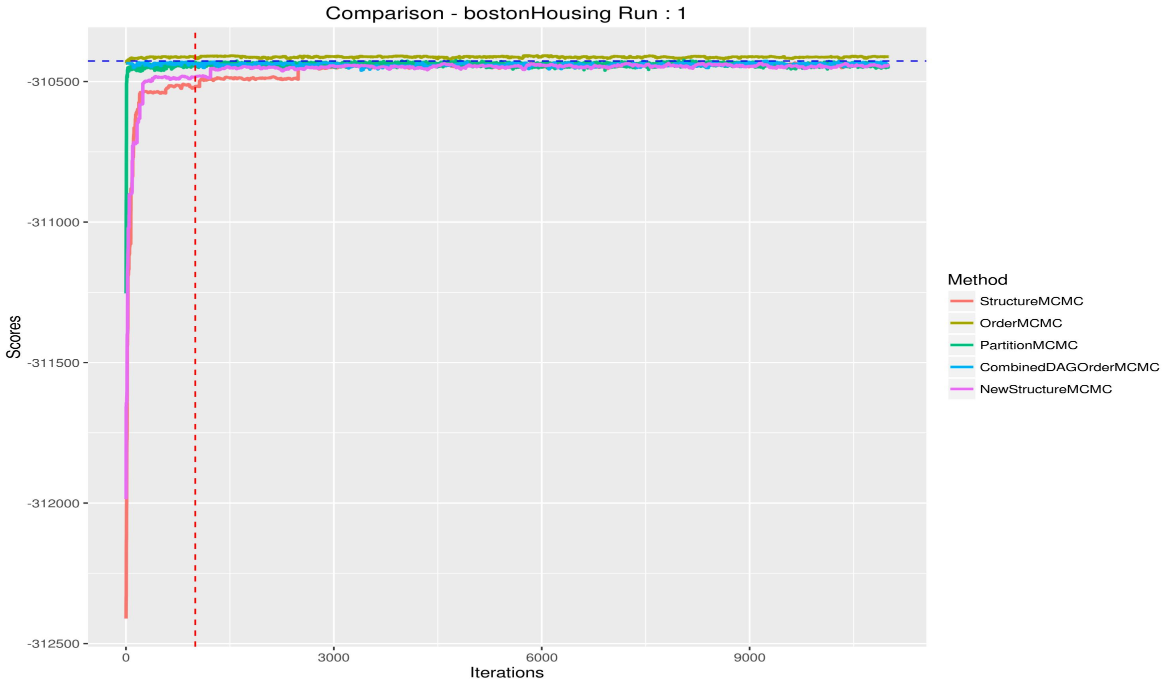


Figure 5.8: Rate of Convergence - Boston Housing

5.2.3 Robustness

We define *robustness* as the nature of the chain to converge irrespective of the starting point in the chain. Theoretically, any MCMC method that satisfies all the convergence criteria mentioned in section 2.2 will converge to the stationary distribution. Analyzing robustness is another way of ensuring that the runs are not biased towards any particular order or structure. Additionally, it also helps us understand if the rate of convergence varies with the starting point because in that case we would need to revisit our condition of detailed balance. Figures 5.5,5.6, 5.7 and 5.8 already show a run of all the methods with starting point chosen at random for all. We conducted another experiment where we ran our method for 10 independent runs on *data5Nodes*. To keep it simple, we chose a much simpler data set *data5Nodes* and collected 100 samples with a burn in of 200 and thinning of 10. It must be noted that we are more interested in the behavior of the chain rather the collected samples. Figure 5.9 shows the results of the runs on Combined DAG Order MCMC. Next we did the same experiment on New Structure MCMC as shown in figure 5.10. Clearly, our method converges irrespective of the starting point with no effect on the rate of convergence. It can be observed that New Structure MCMC does show a little variation depending upon the initiation point. This can be attributed to the reason that it makes use of local moves of Structure MCMC and as a result of which has a tendency to get stuck in a region until the chain is perturbed by a heuristic jump of Combined DAG Order proposal move. Next we study the feature estimation results of our method.

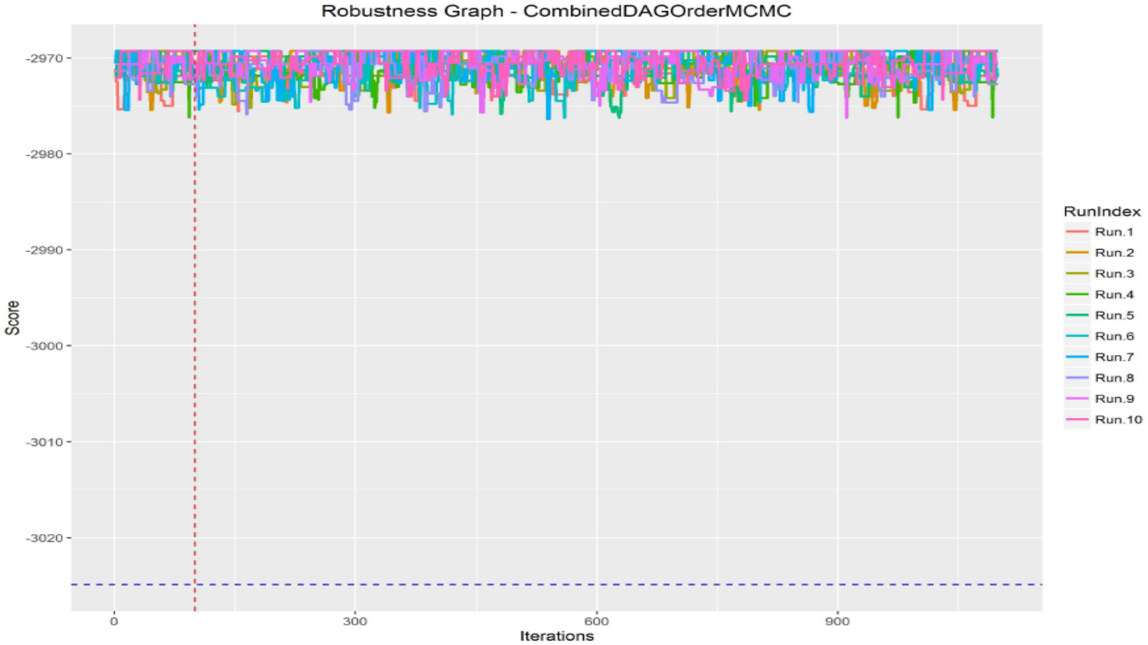


Figure 5.9: Robustness of Combined DAG Order MCMC

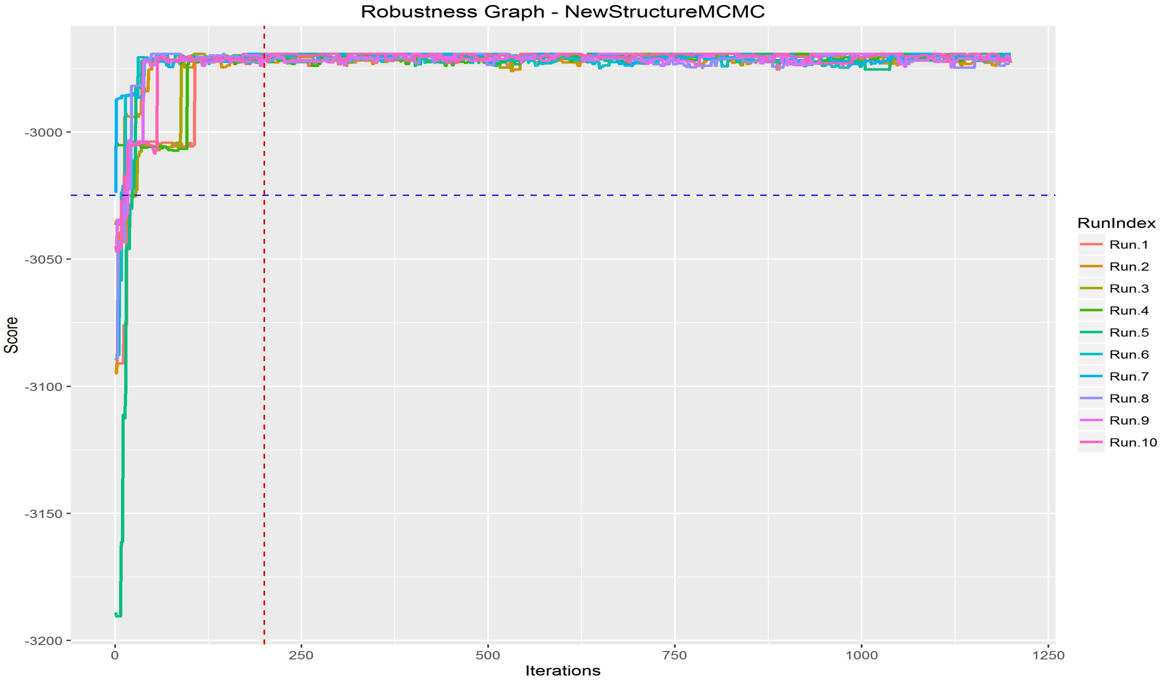


Figure 5.10: Robustness of New Structure MCMC

5.2.4 Feature Estimation

Sometimes prediction is not the only goal, learning about the underlying domain is also important especially in cases where there is no prior knowledge about the domain. Therefore, feature estimates provide an important platform to evaluate our results. To do so, we used two different setups (i) Comparison against true values (ii) Comparison against Partition MCMC and compared values for all the edge features.

Comparison against True values

We evaluated true values of estimates by exhaustively enumerating all the DAGs. This was only possible for a smaller domain and, therefore, we used a synthetically generated data set comprising of 5 nodes. We collected 1000 samples using a burn in of 200 iterations and thinning of 10 iterations. To analyze the results, we plotted true values along the x-axis and feature estimates along Y-axis. Closer the estimates, closer are the points thus obtained to the line with $intercept = 0$ and $slope = 1$. This makes it easy to visually compare the estimates. Figures 5.11 and 5.12 show the results. It can be observed that the points are pretty close to the line as expected. Next we analyze the results of estimates against other methods.

Comparison against Partition MCMC

We have seen that feature estimates are close enough to the true values for all the edges for a simpler model. We are also interested in feature estimate analysis of our methods against the data sets we used. We chose to compare against Partition MCMC. One of the main reasons for doing so is it is one of the very good approaches that has been proposed lately and Order MCMC gives out biased results. Structure MCMC is another candidate for analyzing the results but in order to avoid too much clutter, Partition MCMC is sufficient to analyze feature estimates. We use the same plots as in previous section on comparison against true values for our analysis. We plotted estimates obtained from all the runs on the same plot and thus plotting one graph each for Combined DAG Order v/s Partition MCMC and New Structure MCMC v/s Partition MCMC. Again, we used the same setup of collecting 1000 samples using 200 burn in iterations

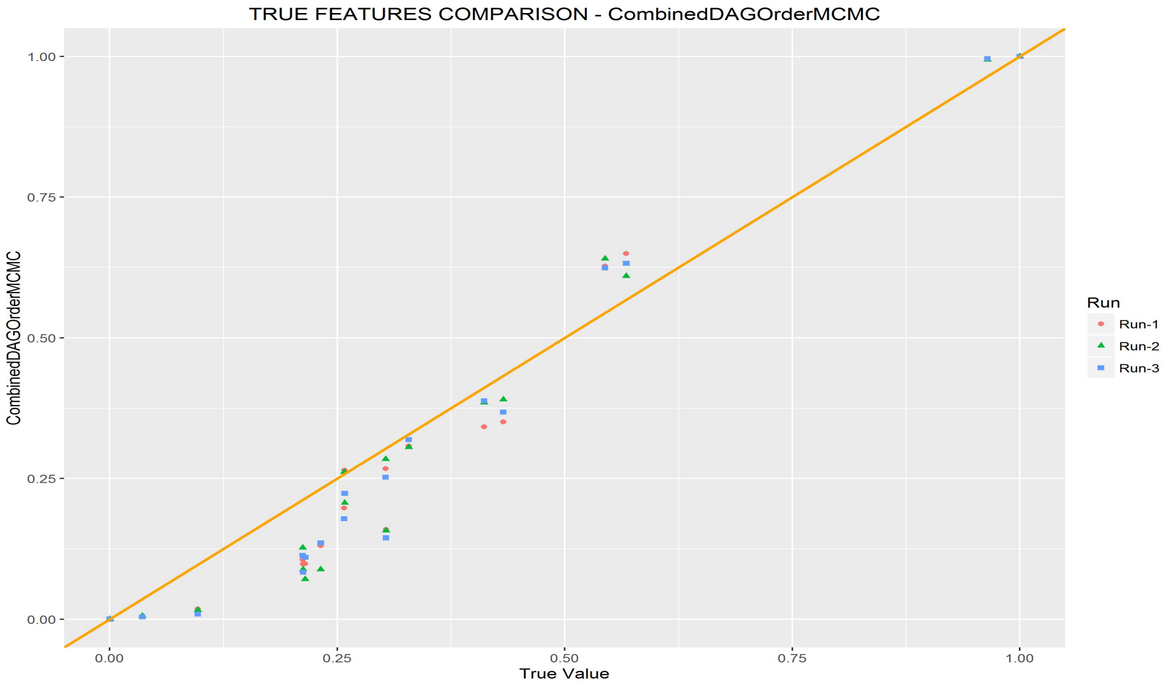


Figure 5.11: Feature Estimate Comparison against true values - Combined DAG Order MCMC

and 10 thinning iterations. Figure 5.13 shows comparison results for Combined DAG Order MCMC and New Structure MCMC. Similarly, we collected the results for data sets (i) Asia (ii) Sachs data sets as shown in figures 5.14 and 5.15. In can be observed that theres an increase in variation in *sachs* data set, which we attribute to a sparsely generated data we used with 100 observations for a 11 node model as we were also interested in the estimation results in case of a sparse data. From the results, it can be inferred that Combined DAG Order MCMC as well New Structure MCMC sound promising approaches and can be looked into further for improvements.

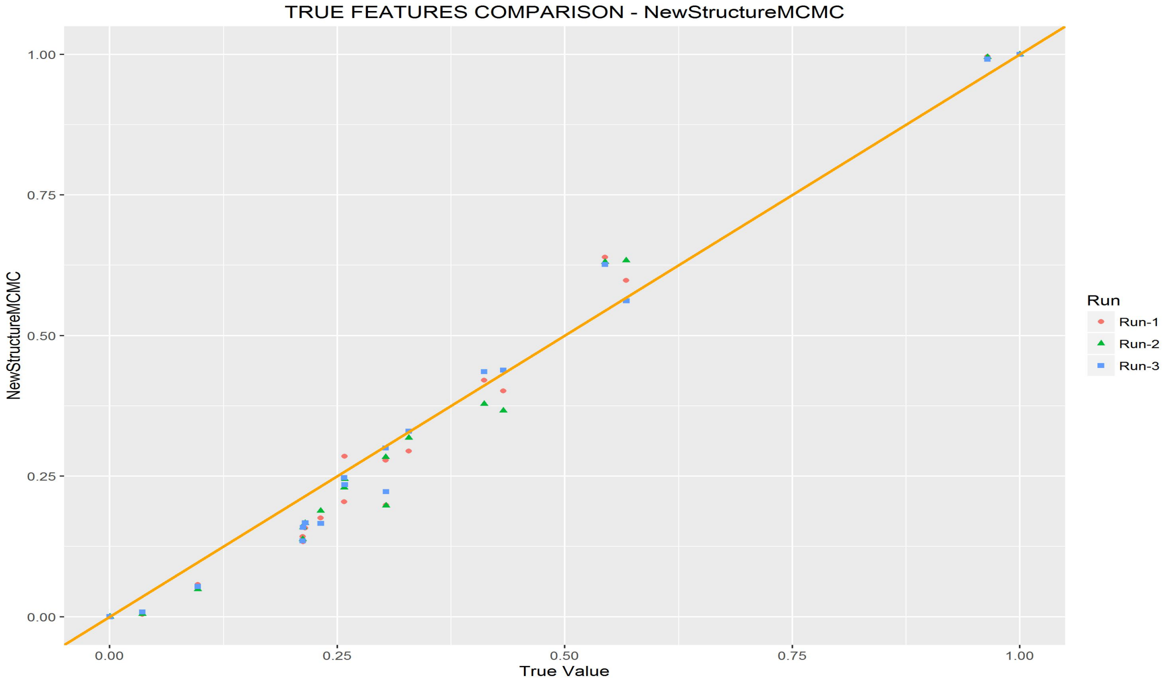
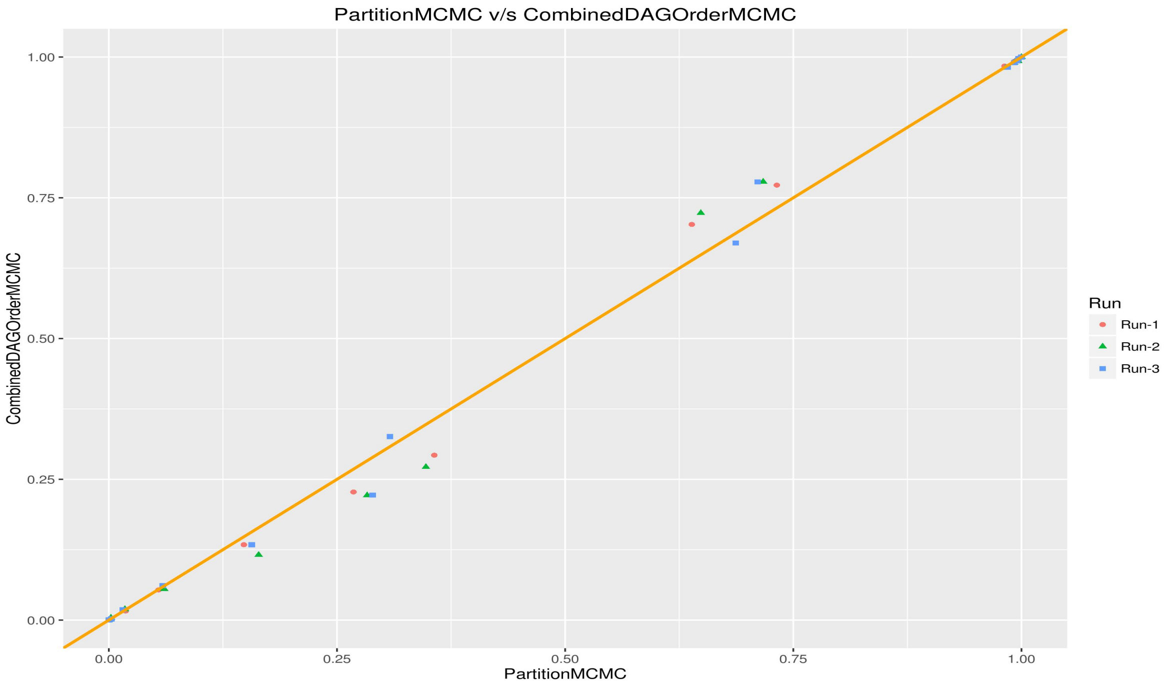
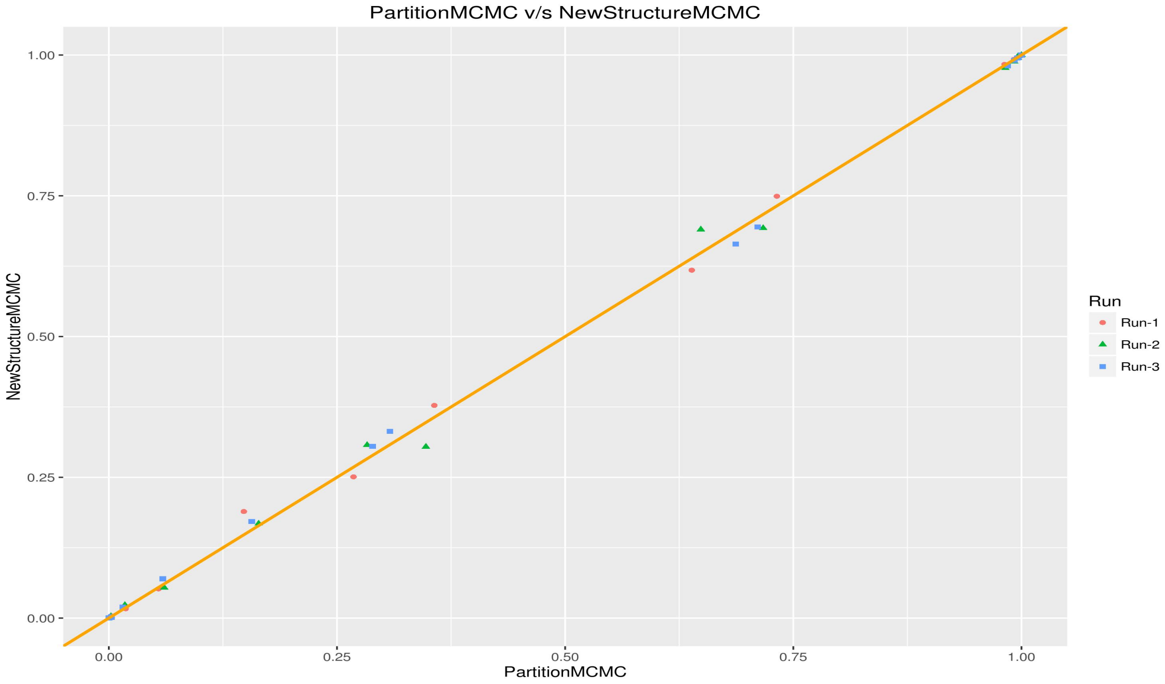


Figure 5.12: Feature Estimate Comparison against true values - New Structure MCMC

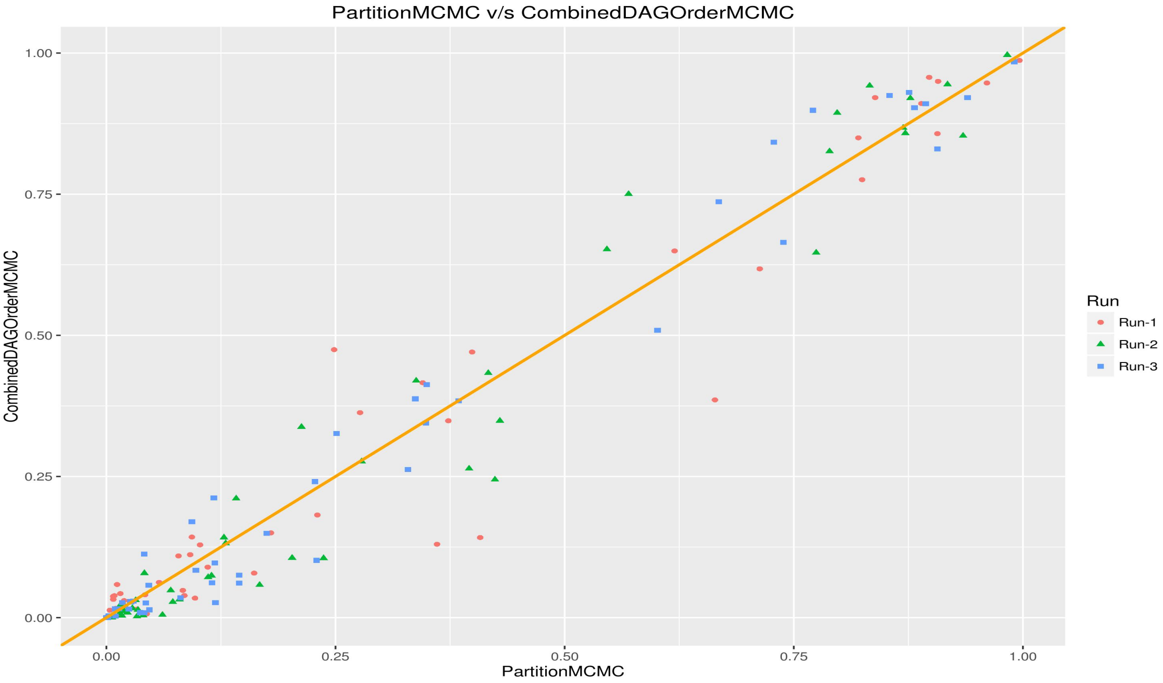


(a) Combined DAG Order MCMC v/s Partition MCMC

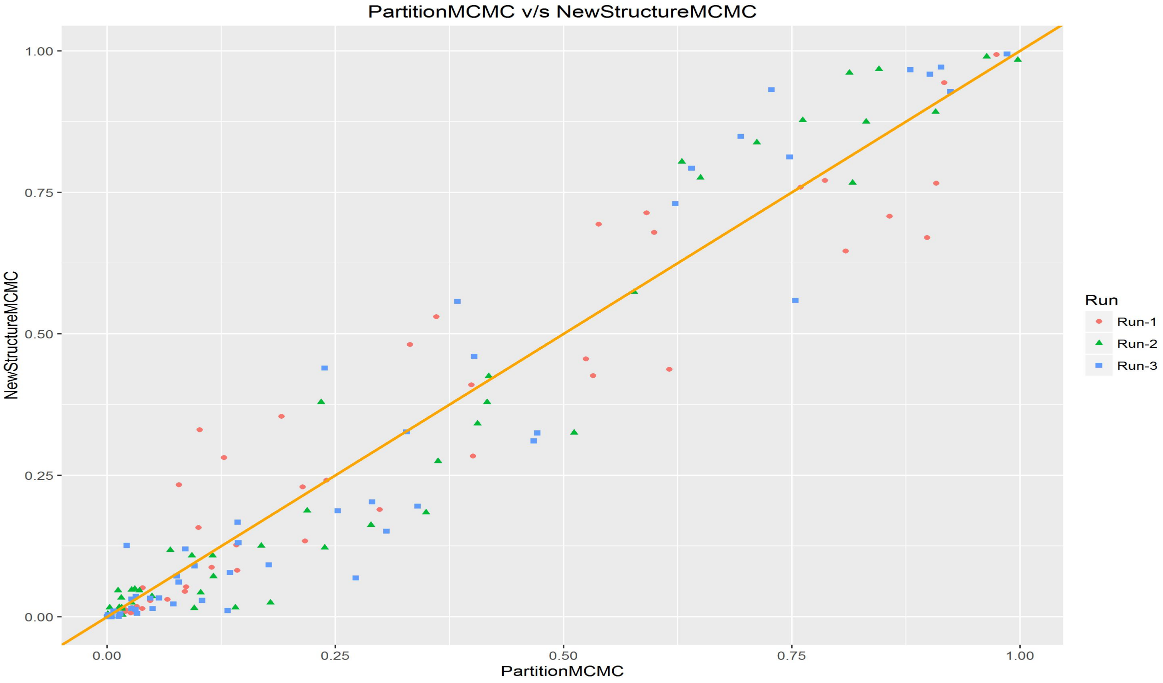


(b) New Structure MCMC v/s Partition MCMC

Figure 5.13: Feature Estimate Comparison - data5Nodes

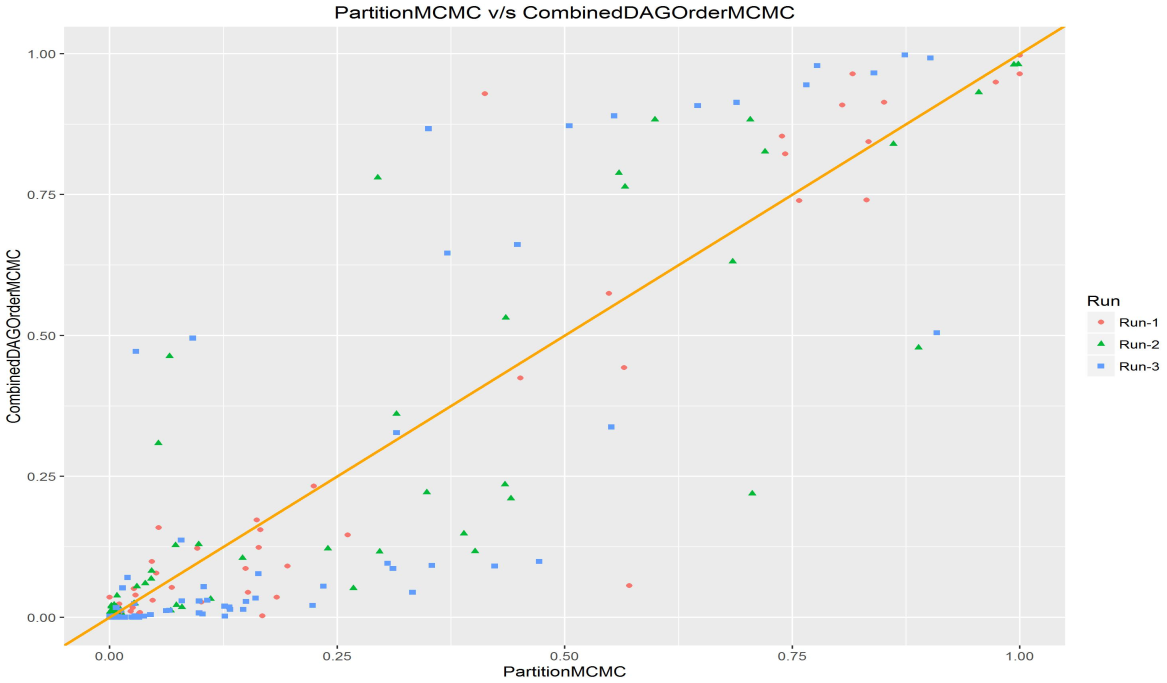


(a) Combined DAG Order MCMC v/s Partition MCMC

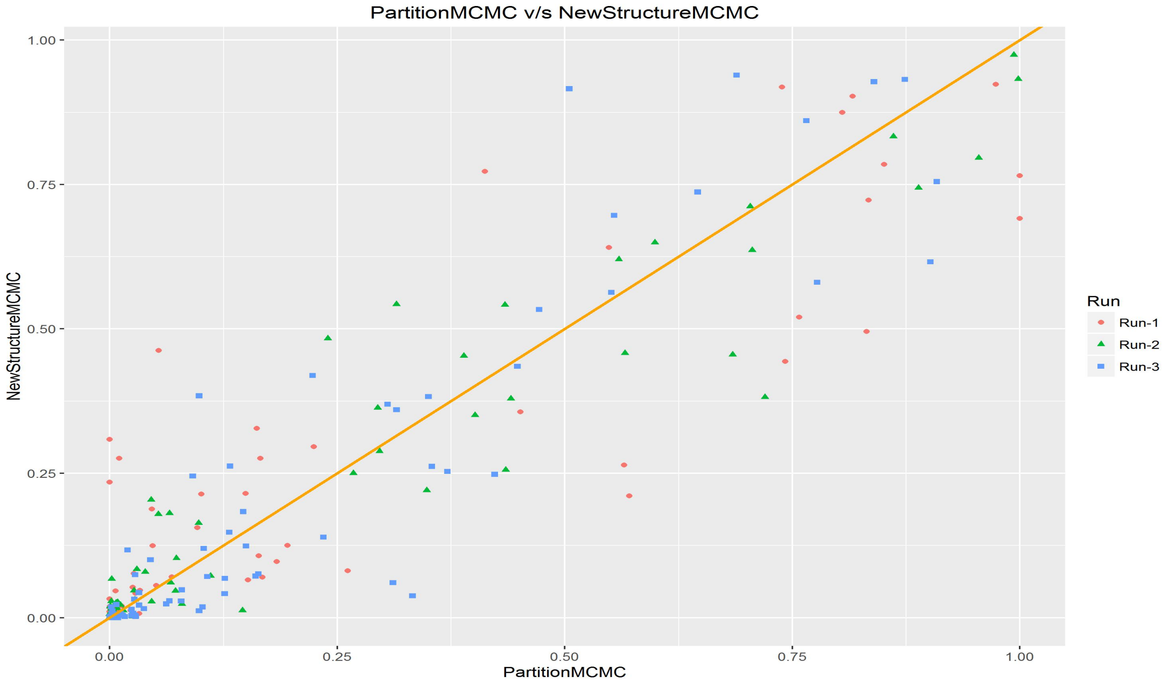


(b) New Structure MCMC v/s Partition MCMC

Figure 5.14: Feature Estimate Comparison - Asia



(a) Combined DAG Order MCMC v/s Partition MCMC



(b) New Structure MCMC v/s Partition MCMC

Figure 5.15: Feature Estimate Comparison - Sachs

Conclusion

We have evaluated both the methods (i) Combined DAG Order MCMC (ii) New Structure MCMC on various criteria against some of the state of the art methods and have found satisfactory results. We compared the quality of sampled DAGs by analyzing the distribution of the score values against some of the well established novel approaches. Then, we compared the rate of convergence and observed that although New Structure MCMC and Structure MCMC are comparatively slower than other methods but they do converge to the stationary distribution with scores of DAGs close the a well known DAG model for the respective domains. Robustness results showed that there was no difference in the behavior of the chains given different random points of initiation. Because New Structure MCMC approach uses local moves of Structure MCMC, slight difference was seen in the rate of convergence. Finally, we evaluated the estimation of all the possible edge features for various data sets and compared against estimates of Partition MCMC and found satisfactory results. All the experimental data presented hints towards this method being a promising approach to tackle the problem and suggest that a more in-depth study is required for improvements. In the next section, we present some of the improvements that can be made to this approach to further strengthen our belief and improve this method.

CHAPTER 6. SUMMARY & FUTURE WORK

In this thesis we present an MCMC technique to learn Bayesian network structure from data. We deal with a known hard problem and chose to use MCMC methodology to sample DAG structures and then use Bayesian model averaging to estimate some of the features of the true underlying graph in order to learn about the domain. We design a Markov chain in the space of DAG structures and use node orderings to make heuristic jumps backed by data in this super-exponential space. This facilitates quicker convergence rate. We diligently calculate the acceptance probability of each and every proposed move for our Metropolis-Hastings sampler to sample DAG structures from a target stationary distribution $P(G|D)$. Previous approaches like Order MCMC and Partition MCMC offer quick convergence rates by sampling orders and partitions respectively. Additionally, Order MCMC adds bias to the results because of the order modular prior assumption. Our method assumes a structure modular prior and samples DAG structures directly without adding any bias. Our approach towards sampling parents of node for a DAG come at a cost of heavy computational steps. In order to bring down the time complexity of our algorithm, we propose to use it in conjunction with local moves of Structure MCMC. This makes the proposal process a three step approach without adding any additional computational costs to the existing process. Additionally, choosing randomly between Structure MCMC proposal moves or Combined DAG Order proposal moves reduces the time complexity as local structure moves are fairly easy to compute. We also presented some of the computation tricks we have used in this work. However, there is still a lot of scope in this area. One possibility would be to use the tricks used by Friedman and Koller (2003) in their paper, where they do not evaluate any parent sets any further when addition of a node significantly reduces the score. As is indicated by the sampling results, there can be a lot of DAG structures that have the same score and since we are dealing with logarithmic values, even a decrease in a score by a value of 10 implies a change in actual probability score by a factor of 2^{10}, which is a significant value. For combining it with Structure MCMC, we sample an order uniformly that a specific DAG is consistent with. We have already seen that in combination with Structure MCMC, we get good results with a significant reduction in overall run time as compared to running Combined DAG Order MCMC alone and a faster convergence rate as compared to running Structure MCMC alone. If we could make a calculated move according to likelihood of order given DAG structure and data, we could potentially improve the rate of convergence of Structure MCMC.

In the end, we would conclude that the experimental results suggest that both the methods offer a possibility of arriving at a potential technique to learn Bayesian network structure given data and there is a lot of scope for further improvements to enhance this method.

BIBLIOGRAPHY

Andrieu, C., Freitas, N. D., and et al. (2003). An introduction to mcmc for machine learning.

Buntine, W. (1991). Theory refinement on bayesian networks. In *Proceedings of the Seventh Conference (1991) on Uncertainty in Artificial Intelligence*, pages 52–60, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Colombo, D. and Maathuis, M. H. (2014). Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782.

Cooper, G. F. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9(4):309–347.

Daly, R., Shen, Q., and Aitken, S. (2011). Learning bayesian networks: approaches and issues. *The knowledge engineering review*, 26(02):99–157.

Eaton, D. and Murphy, K. (2012). Bayesian structure learning using dynamic programming and mcmc. *arXiv preprint arXiv:1206.5247*.

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799–805.

Friedman, N. and Koller, D. (2003). Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine Learning*, 50(1):95–125.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620.

Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (1996 (ISBN: 0-412-05551-1)). *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London.

This book thoroughly summarizes the uses of MCMC in Bayesian analysis. It is a core book for Bayesian studies.

Giudici, P. and Castelo, R. (2003). Improving markov chain monte carlo model search for data mining. *Machine Learning*, 50(1-2):127–158.

Grzegorczyk, M. and Husmeier, D. (2008). Improving the structure mcmc sampler for bayesian networks by introducing a new edge reversal move. *Machine Learning*, 71(2):265–305.

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

Heckerman, D. (1995). A bayesian approach to learning causal networks. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pages 285–295, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Heckerman, D. (1996). A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models.

Hill, S. M., Lu, Y., Molina, J., Heiser, L. M., Spellman, P. T., Speed, T. P., Gray, J. W., Mills, G. B., and Mukherjee, S. (2012). Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics*, 28(21):2804–2810.

Husmeier, D. (2003). Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. *Bioinformatics*, 19(17):2271–2282.

Kalisch, M. and Bühlmann, P. (2007). Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *J. Mach. Learn. Res.*, 8:613–636.

Koivisto, M. and Sood, K. (2004). Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Research*, 5(May):549–573.

Kuipers, J. and Moffa, G. (2015a). Partition mcmc for inference on acyclic digraphs. https://github.com/annlia/partitionMCMC.

Kuipers, J. and Moffa, G. (2015b). Uniform random generation of large acyclic digraphs. *Statistics and Computing*, 25(2):227–242.

Kuipers, J. and Moffa, G. (2016). Partition mcmc for inference on acyclic digraphs. *Journal of the American Statistical Association*.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224.

Lichman, M. (2013). UCI machine learning repository.

Madigan, D., York, J., and Allard, D. (1995). Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Mukherjee, S. and Speed, T. P. (2008). Network inference using informative priors. *Proceedings of the National Academy of Sciences*, 105(38):14313–14318.

Rau, A., Jaffrézic, F., Foulley, J.-L., and Doerge, R. W. (2012). Reverse engineering gene regulatory networks using approximate bayesian computation. *Statistics and Computing*, 22(6):1257–1271.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529.

Scutari, M. (2010). Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22.

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*. MIT press, 2nd edition.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Mach. Learn.*, 65(1):31–78.