Master's thesis

Master's Programme in Theoretical and Computational Methods

Sampling Directed Acyclic Graphs Using Dynamic Parallel Tempering Schemes

Daniele Nikzad

7th November 2024

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

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

University of Helsinki

Faculty of Science

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

00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

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

Tekijä — Författare — Author

Daniele Nikzad

Työn nimi — Arbetets titel — Title

Sampling Directed Acyclic Graphs Using Dynamic Parallel Tempering Schemes

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	7th November 2024	87

Tiivistelmä — Referat — Abstract

Bayesian networks are popular graphical models that describe quantitative and qualitative relations between variables starting from data. The Bayesian approach is used to give a prior probability to each directed acyclic graph to favor specific properties of a network.

The structure Markov chain Monte Carlo (MCMC) is a widely used algorithm that estimates the posterior distribution of directed acyclic graphs (DAGs). It may use different types of moves to propose new DAGs, which are accepted according to a Metropolis–Hastings ratio. In this Master's thesis, different techniques that increase the performance of this MCMC method are compared and combined to obtain a state-of-the-art structure MCMC algorithm for sampling DAGs. Experiments are conducted to show the performance improvement achieved by the above-mentioned sampler.

Parallel tempering is widely used in MCMC methods to allow a better exploration of the target distribution, helping the MCMC runs to cross low-probability regions of multimodal distributions. It is implemented by constructing communicating chains that sample from distributions at different temperatures. Temperatures are scalar values that define how much the target distribution affects the distribution from which the associated chain samples. In the experiments, different parallel tempering schemes are implemented and compared, including the novel deterministic even-odd algorithm (DEO) that has been proven to outperform other popular schemes. A new tuning routine, divided into two phases, has been introduced to reduce the computational cost of finding temperatures that guarantee good communication between chains. A dynamic tuning scheme, which has never been proposed before, may be enabled by not interrupting the tuning algorithm to further increase communication between chains.

ACM Computing Classification System (CCS):
Computing methodologies → Machine learning → Machine learning approaches → Learning in probabilistic graphical models → Bayesian network models
Mathematics of computing → Probability and statistics → Probabilistic reasoning algorithms → Markov-chain Monte Carlo methods

Avainsanat — Nyckelord — Keywords

Bayesian networks, structure MCMC, parallel tempering

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

Discovering the quantitative and qualitative interaction between variables is an important topic in computational statistics. Bayesian networks (BN) are popular probabilistic graphical models that are used to infer conditional relationships between variables. Bayesian networks are represented by directed acyclic graphs (DAGs), where the edges show the presence of direct causalities and dependencies between the nodes. Conditional probability tables are used for each node to describe the quantitative relationships among variables and the joint probability distribution of the variables of a network is computed by exploiting the conditional independencies between the variables. The Bayesian approach implies the implementation of a prior probability for each DAG, which can penalize or favor the sparsity of a DAG. Structure discovery relies only on the available data, but constraints regarding relationships between nodes can be given.

Friedman and Koller [1] argue that finding only the most probable DAG may result in loss of important information about the network, especially when the available data points are low in number, compared to the number of nodes of the network N. Frequently, in Bayesian networks, a Bayesian model averaging over different DAGs is performed [2]. Consequently, the degree of belief of the presence of an edge is described by a probability value, which is called posterior probability of the edge.

The posterior distribution of a DAG is represented as a product of conditional probabilities of the variables, given their parents set. This property, which is called structure modularity, facilitates structure discovery using Markov chain Monte Carlo (MCMC) methods.

Madigan and York [3] developed an MCMC algorithm for sampling DAGs (frequently denoted as structure MCMC), where the new DAG is proposed by adding or removing a single edge from the current DAG of the MCMC. A Metropolis–Hastings ratio [4, 5] is computed to assess the acceptance ratio of the proposed DAG. Giudici and Castelo [6] improved the above-mentioned DAGs sampler by introducing a new

move that allows the reversal of an edge in a single step. However, this new move does not help to overcome the limitations of the structure MCMC. The proposed DAG differs from the current DAG of the MCMC run by one single edge. Since the space of DAGs grows super-exponentially in the number of variables and the likelihood term tends to favor specific DAGs, the posterior distribution may present extended low-probability regions that this sampler is not able to cross efficiently.

This problem has been partially solved by the introduction of new structure MCMC moves. The New Edge Reversal (REV) move, introduced by Grzegorczyk and Husmeier [7], generates a proposed DAG by reversing a randomly selected edge and by sampling new parents sets for the nodes at the extremity of the edge. These parents sets are chosen among those that preserve the acyclicity of the DAG. The Markov Blanket Resampling (MBR) move, described by Su and Borsuk [8], proposes new DAGs by sampling a new Markov blanket of a randomly selected node. A structure MCMC that can perform a combination of simple moves (add, remove, reverse), REV moves, and MBR moves, results in a faster exploration of the space of DAGs.

When the number of nodes of the network increases and the posterior distribution of DAGs is multimodal, even this type of sampler may fail to sample DAGs according to the posterior distribution. Therefore, further implementations to the structure MCMC are needed to obtain a sampler that can more easily cross low-probability regions of the posterior distribution.

Parallel tempering is a powerful tool that is widely applied to MCMC algorithms [9, 10, 11, 12] as it ensures the correct exploration of complex target distributions. It relies on different MCMC runs that sample from distributions at different "temperatures" and communicate with each other. Colder chains sample from distributions that do not present sharp local maxima, making these distributions more easily explorable than the target distribution. Starting from the scheme proposed by Geyer [9], several improvements have been proposed and tuning algorithms for the temperatures have been introduced to obtain good communication between chains.

Syed et al. [10] describe comprehensively the state-of-the-art deterministic even-odd (DEO) swap parallel tempering scheme, which was introduced by Okabe [11]. The DEO scheme is non-reversible and deterministically selects the even or odd indexes from which the parallel tempering swap is proposed. The DEO scheme has been proven to outperform reversible parallel tempering algorithms that involve random walks [10], ensuring better communication between the chain that samples

from the target distribution (posterior distribution), and the chain that samples from the reference distribution (prior distribution).

In this Master's thesis, the structure MCMC, provided with a hybrid exploration kernel (the sampler can perform simple moves, REV and MBR moves), has been combined with the DEO parallel tempering scheme to ensure a correct exploration of the posterior distribution and to obtain the state-of-the-art structure MCMC for Bayesian structure discovery.

The Master's thesis is organized into six chapters. Chapter 2 provides a detailed explanation of the theory of Bayesian networks and the theoretical foundations leading to the decomposition of the posterior distribution. Chapter 3 presents, in chronological order, the various MCMC structure moves used to propose new DAGs, and how these moves are integrated into a single DAG sampler. Chapter 4 discusses the theory behind tuning in parallel tempering and examines the theoretical performance of popular parallel tempering schemes. Chapters 5 and 6 present the experimental results and the conclusions, respectively.

2. Bayesian networks

In this chapter, the main features of Bayesian networks will be presented after a general introduction to Bayesian statistics.

2.1 The Bayesian approach

Bayesian statistics is a branch of statistics that describes in mathematical terms the concept of epistemic uncertainty (which derives from lack of knowledge). It follows that probability expresses the degree of belief of an event. Bayesian statistics relies on Bayes' formula that can be written as

$$p(\theta|D, M) = \frac{p(D|\theta, M)p(\theta|M)}{p(D|M)} \quad , \tag{2.1}$$

where θ is the model parameter, D is the observed data and M is the model that also incorporates the background information.

2.1.1 Prior distribution and likelihood function

The prior distribution $p(\theta|M)$ represents the knowledge of the parameter before the observation took place. Different priors can be used in different contexts, and the choice depends on the amount of information we want to include in the model. It is possible to distinguish between informative priors and non-informative priors and the concept of subjectivity derives from the possibility of choosing among different priors.

The term $p(D|\theta, M)$ is called likelihood. It represents the probability of obtaining the observed data given the model parameter and the chosen statistical model. The observations directly influence the likelihood, and, consequently, the more data points are incorporated into the model, the more the shape of the posterior is affected by the likelihood. In frequentist statistics, the likelihood is the only term analyzed. Maximum Likelihood Estimator is (MLE) is widely used in frequentist statistics as an estimator of the model parameters (the prior information is not taken into account). When the

number of data points is small, this type of analysis could lead to results that are not accurate. The Bayesian approach, which includes prior information, is beneficial when the number of data points is limited. However, when the number of data points is extremely large, frequentist statistics and Bayesian statistics tend to converge to the same model parameter estimation since the likelihood term is strong and the prior term loses its influence on the posterior distribution.

2.1.2 Posterior distribution and frequently used estimators

The probability function $p(\theta|D, M)$ of the model parameter conditioned on the observed data and on the chosen model is called posterior distribution. It represents the knowledge about the parameter after one or more observations have taken place. The closed form solution of the posterior distribution is usually extremely challenging to compute. Several methods can be applied to approximate the posterior distribution depending on the nature of the model. Markov chain Monte Carlo (MCMC) methods and variational inference (VI) are the most popular.

Among MCMC methods, the Metropolis–Hastings algorithm [4, 5] is still widely used due to the lack of restrictions needed on the model. It can be applied easily on both discrete and continuous distributions. No U-Turn Hamiltonian Monte Carlo (NUTS) [13] samplers are used when the parameter is a differentiable function to obtain faster convergence to the target distribution, while the Gibbs sampler [14] is preferred when the parameter is multidimensional. This latter method implies that each component of the K-dimensional parameter is sampled iteratively while the other components are kept to the same value, leading to an acceptance ratio of the proposed state equal to 1. However, when the model parameter has a large number of dimensions K, it might not be recommended to use Gibbs sampling since, to have a single update, K sampling steps are needed. MCMC methods are widely used, in general, because they guarantee asymptotical convergence, even though they require a high computational cost.

Variational inference is less computationally heavy, but approximations on the nature of the target distributions are needed. Coordinate Ascent Variational Inference (CAVI) algorithm [15] is a popular VI algorithm. There are similarities between Gibbs sampling and CAVI. In the latter model, the mean-field factors are iteratively updated, climbing the evidence lower bound (ELBO) to a local maxima. In the experiments Markov chain Monte Carlo (MCMC) methods are used to approximate the posterior distribution. The algorithms are shown in detail in the following chapters.

2.1.3 Marginal likelihood

The term at the denominator in Equation 2.1 is called marginal likelihood or normalization constant. If θ is a continuous random variable, the marginal likelihood $p(D|M)$ can be written as

$$p(D|M) = \int p(D|\theta, M)p(\theta|M)d\theta \quad . \tag{2.2}$$

If θ is a discrete random variable and Ω is the space that contains all and only the possible states of θ, then the formula for the marginal likelihood becomes

$$p(D|M) = \sum_{i=1}^{|\Omega|} p(D|\theta_i, M)p(\theta_i|M) \quad , \tag{2.3}$$

where θ_i refers to a state in the space Ω, and $|\Omega|$ denotes the cardinality of the set of possible states in the space. However, computing the marginal likelihood poses challenges similar to those encountered when calculating the posterior distribution. Obtaining a closed-form solution is often difficult due to the complexity of the summations (or integrals, in the continuous case). Thus, simplifying assumptions on the prior distribution of the parameters are frequently made, and approximation techniques are used to make the computation feasible.

2.1.4 Advantages of the Bayesian approach

In the Bayesian approach it is possible to take into account different models simultaneously. In this case, a prior probability must be defined for each of the $|\mathcal{M}|$ models (where $\mathcal{M}$ is the space of the models). In Bayesian networks, $\mathcal{M}$ represents the space of the possible structures of a network. The prior probability distribution satisfies the condition $\sum_{i=1}^{|\mathcal{M}|} p(M_i) = 1$ and each probability should be non-negative. Hence the formula for the posterior distribution of a model is

$$p(M_m|D) = \frac{p(D|M_m)p(M_m)}{\sum_{m=1}^{|\mathcal{M}|} p(D|M_m)p(M_m)} \quad . \tag{2.4}$$

An advantage of the Bayesian approach consists in the ease of updating the posterior distribution, after new observations are added to the model. Let us assume, for example, that D_1 is an event that already occurred, while D_2 is the new observation. The posterior distribution becomes

$$p(\theta|D_1, D_2, M) = \frac{p(D_2|D_1, \theta, M)p(\theta|D_1, M)}{p(D_2|D_1, M)} \quad . \tag{2.5}$$

The likelihood term can be expressed in different ways, based on the dependence of the observations. Another advantage of the Bayesian approach derives from the ease of implementation of hierarchical models. In hierarchical models, the priors are conditionally dependent on hyperparameters η. Hence, we can write the prior as $p(\theta|\eta, M)$ and the hyperprior as $p(\eta|M)$.

2.2 Properties of Bayesian networks

Bayesian networks model the conditional dependencies and causalities among variables, which are represented by the direct edges in directed acyclic graphs (DAGs), where the nodes of the DAGs are the variables taken into account.

2.2.1 Directed acyclic graphs

The notation used in this Master's thesis is the one suggested by Neapolitan [16]. A graph M can be described by the set of nodes V and the set E of directed edges that connect some nodes. The set E is commonly expressed as a collection of ordered pairs of nodes. If there is an edge connecting the node X_i to the node X_j we write $(X_i, X_j) \in E$. In this case, node X_i is a parent of node X_j, while node X_j is a child of node X_i. The set (V, E) is a directed acyclic graph (DAG) if the nodes are connected in a way that no cycles are formed. The set that contains all and only the parents of X_i is called parents set of X_i. It contains all the nodes in a graph M that have an edge pointing to X_i. A popular notation for the parents set of X_i is π_i. If X_k can be reached following the direct edges of a DAG M, starting from X_i, then X_k is a descendant of X_i. In Bayesian networks, the edges of a DAG represent the conditional relationships among variables. Figure 2.1 is a graphical visualization of the Asia benchmark network [17], from which a synthetic dataset has been sampled to conduct experiments in Chapter 5.

2.2.2 Markov condition

From the above assumptions, it is straightforward that the direct edges connecting two nodes represent an immediate dependence. However, the absence of edges connecting two nodes does not imply the lack of causality between the variables, since an indirect causality is present if node X_k is a descendant of X_i. A statement about the independence of 2 sets of variables given a third one is given by the d-separation (direct

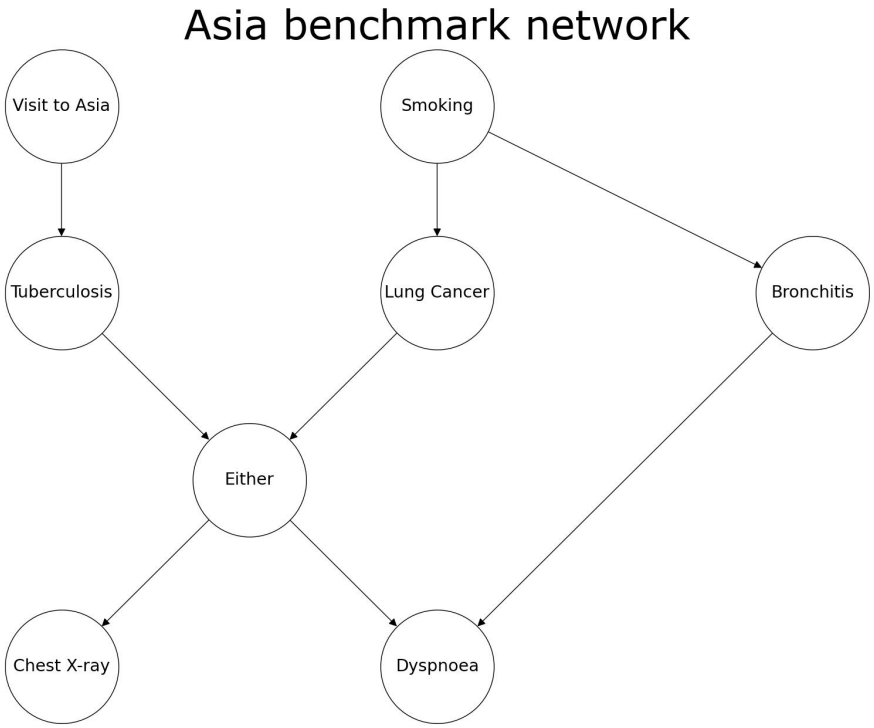

Figure 2.1: Visualization of the Asia benchmark network. The edges represent the direct causalities among the variables.

separation) definition [18].

Definition 1 (d-separation) *Given three disjoint sets of nodes A, B, C of a DAG M, then A and B are d-separated by C if every possible path connecting any node in A and any node in B is blocked by a node that belongs to C. D-separation implies conditional independence on C.*

We can now define the Markov condition as a special case of d-separation [16].

Definition 2 (Markov condition) *Given a graph M, its nodes X_i and their joint distribution $p(\cdot)$. If the variables X_i are conditionally independent on all the non-descendants, given the parents sets of X_i, then the Markov condition is satisfied.*

The Markov blanket of a node X_i contains the smallest set of nodes such that, when X_i is conditioned on them, X_i is conditionally independent on all the other nodes. Theorem 1 shows how to obtain the Markov blanket of a node X_i in a DAG M [16].

Theorem 1 (Markov blanket) *Given a graph M and the joint probability $p(\cdot)$ of the nodes X_i of M, that satisfy the Markov condition, then the Markov blanket of X_i*

is the set of nodes containing the children of X_i, the parents of X_i, and the parents of the children of X_i.

For example, in the Asia benchmark network shown in Figure 2.1, the Markov blanket of Lung cancer variable, is the set containing Smoking, Tuberculosis and Either variables.

2.2.3 Structure modularity

Structure modularity is a consequence of the Markov condition and its definition can be found in the theorem below [16].

Theorem 2 (Structure modularity) *Given a graph M, if the Markov condition is satisfied, then the joint probability distribution $p(\cdot)$ of the variables X_i forming M can be expressed as the product of the probability distributions of X_i conditioned on their parents set.*

Hence the joint probability formula $p(\cdot)$ of a DAG M can be written as

$$p(X_1, ..., X_N) = \prod_{i=1}^{N} p(X_i|\pi_i) \quad , \tag{2.6}$$

where N is the number of nodes of the network. Different DAGs may share the same conditional independence relations among the nodes X_i. This concept introduces the definition of equivalence class. DAGs that share the same skeleton (generated by removing the direction of the edges) and that share the same v-structures belong to the same equivalence class [19]. The v-structures are formed by three nodes X_i, X_j, X_k. In this configuration, there are two nodes X_i and X_j, that have an edge that points at the node X_k while there is no direct edge between X_i and X_j. In the DAG shown in Figure 2.1, for example, a v-structure is represented by the nodes Tubercolosis, Lung cancer, and Either.

Only qualitative features about the structure of Bayesian networks have been discussed until now. The quantitative features are expressed by the parameterization of $\boldsymbol{\theta}_M$ associated to the probabilities of the states of the variables [20]. In a discrete Bayesian network, $\boldsymbol{\theta}_M$ usually consists of multinomial distributions described by $\boldsymbol{\theta}_{X_i|\boldsymbol{u}}$ for each node and for each $\boldsymbol{u}$ among the possible combinations of π_i. The computation becomes more complex, and a variance parameter is added, when continuous variables are taken into account and a linear-Gaussian approximation of the distribution is used (Gaussian Bayesian networks).

2.3 Priors in Bayesian networks

Discovering the posterior distribution of DAGs, given a dataset D, is the main objective in Bayesian networks analysis. We are using the plural (DAGs) since finding the DAG that maximizes the posterior distribution would result in loss of information in most of the circumstances. In the theoretical scenario where the number of data points tends to infinite, the likelihood term becomes extremely sharp. Therefore, knowing only the highest scoring DAGs would result in less loss of information. In realistic scenarios, where the number of data points is limited, a large set of DAGs has to be taken into account. This concept was already introduced in Section 2.1.4 (Equation 2.4).

A prior distribution for all possible DAGs has to be chosen. The most straightforward prior is the uniform prior, where all DAGs have the same prior probability. Another frequently used prior is the sparse prior. Assuming structure modularity, the prior for the whole DAG is [1]

$$p(M) \propto \prod_{i=1}^{N} \binom{N-1}{|\pi_i|}^{-1} \quad , \tag{2.7}$$

where N is the number of nodes and $|\pi_i|$ is the number of parents in the parents set π_i. In the experiments, this latter prior is selected since it allows a good exploration of the space of the DAGs. Priors that favor sparsity have been proven to perform better than the uniform prior in different settings [21].

2.4 Structure modularity of the posterior distribution

For a correct inference of the posterior distribution, the prior must satisfy some particular conditions [1].

Definition 3 (Global parameter independence) *Assuming that $\boldsymbol{\theta}_{X_i|\boldsymbol{\pi_i}}$ is the parameterization of X_i, conditioned on the chosen parents set. Global parameter independence is satisfied if*

$$p(\boldsymbol{\theta}_M|M) = \prod_{i=1}^{N} p(\boldsymbol{\theta}_{X_i|\pi_i}|M) \quad . \tag{2.8}$$

Definition 4 (Parameter modularity) *Given two different DAGs M and M', where the node X_i has the same parents set π_i for both M and M', parameter modularity is satisfied if*

$$p(\boldsymbol{\theta}_{X_i|\pi_i}|M) = p(\boldsymbol{\theta}_{X_i|\pi_i}|M') \quad . \tag{2.9}$$

Consequently, the marginal likelihood term (Equation 2.2) can be expressed as

$$p(D|M) = \int p(D|\boldsymbol{\theta}_M, M)p(\boldsymbol{\theta}_M|M)d\boldsymbol{\theta}_M \quad . \tag{2.10}$$

The above-mentioned conditions, combined with structure modularity (Theorem 2) are needed to obtain a convenient formula for the posterior distribution [1].

Theorem 3 (Posterior distribution decomposition) *If D is complete and $p(M)$ satisfies parameter independence, parameter modularity and structure modularity, then the posterior distribution decomposes as*

$$p(M|D) \propto p(M, D) = \prod_{i=1}^{N} \exp(\Psi[X_i, \pi_i|D]) \quad . \tag{2.11}$$

Using log probabilities, we obtain the log marginal likelihood as the sum of the log local scores of each node $\sum_{i=1}^{N} \Psi[X_i, \pi_i|D]$. The log local score $\Psi[X_i, \pi_i|D]$ is the sum of the log marginal likelihood term of the node X_i and of the log prior term of the node X_i and it can be computed exactly for linear-Guassian models [22] and for multinomial models [23].

2.5 BDeu metric

In the experiments, BDeu score metric is used to compute the log marginal likelihood, since the experiments are performed using discrete datasets. The datasets are

described in detail in Section 5.2. Starting from the BD (Bayesian Dirichlet) metric [23]

$$p(D, M) = P(M) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})} \quad , \tag{2.12}$$

where r_i is the number of the possible states of X_i, q_i is the number of possible states of π_i, $n = \sum_{ijk} n_{ijk}$ is the number of points observed in D and α_{ijk} are the Dirichlet exponents used to give information about the prior using imaginary samples. More precisely, n_{ijk} is the number of times the variable X_i has value j when the parents set configuration of X_i corresponds to the index k. Let us now introduce the concept of likelihood equivalence [24].

Definition 5 (Likelihood equivalence) *Given two network structures M_1 and M_2 such that $p(M_1) > 0$ and $p(M_2) > 0$, if M_1 and M_2 are equivalent, then $p(\boldsymbol{\theta}|M_1) = p(\boldsymbol{\theta}|M_2)$.*

Therefore, $\alpha_{ijk} = \alpha \cdot p(X_i = j, \pi_i = k|M)$. By accepting the assumption of using uninformative hyperparameters, we get $\alpha_{ijk} = \frac{\alpha}{r_i q_i}$ [24]. BDeu metric is finally obtained by applying the logarithm to the marginal likelihood formula. BDeu score is widely used since it implies that equivalent DAGs have the same score.

Other BD scores have been parameterized. For example, by setting $\alpha_{ijk} = 1$, we obtain a new BD score that is called K2 [23]. If $\alpha = 0.5$ the model is called BDJ where the J refers to Jeffrey's prior [25]. Another model, called BDs (Bayesian Dirichlet sparse) [26] is obtained by setting $\alpha = \frac{1}{\tilde{q}_i r_i}$ where $\tilde{q}_i$ refers to the cardinality of the set of parents set π_i, taking only into account the parents sets where $n_{ij} > 0$.

3. Sampling DAGs to estimate the posterior distribution

Bayesian model discovery focuses on finding a degree of belief for each DAG. It is a difficult task since the space of DAGs grows super-exponentially with the number of nodes [27]. The equation for the number of possible states, depending on the number of nodes of the network, is obtained by the recurrence

$$|\mathcal{M}(N)| = \sum_{k=1}^{N} (-1)^{k+1} \binom{N}{k} 2^{k(N-k)} |\mathcal{M}(N-k)| \quad , \tag{3.1}$$

where N is the number of nodes of the network and $|\mathcal{M}(N)|$ is the cardinality of the space of possible DAGs, when the number of nodes of the newtwork is N. The computation of an exact posterior may be too challenging, since the problem of computing the normalization constant is #P-hard. The space of DAGs can be reduced by assigning a maximum number of parents to each node, reducing the computational cost and avoiding overfitting.

Starting from Equation 2.4, that describes the posterior distribution of $|\mathcal{M}|$ models, it is possible to derive the posterior distribution of a quantity of interest, by using Bayesian model averaging [3]. Therefore,

$$p(\Delta|D) = \sum_{m=1}^{|\mathcal{M}|} p(\Delta|M_m, D) p(M_m|D) \quad . \tag{3.2}$$

If algorithms that sample DAGs according to the posterior distribution are used, it is possible to simplify the above shown equation [3]. Given $\mathcal{M}$ representing the class of possible models, a set of sampled DAGs defined in $\mathcal{M}$, can be expressed by the function $M(t)$ with $0 < t \leq T$ where T is the total number of DAGs sampled. A consistent empirical estimator of $\mathbb{E}(M)$ [3] is represented by

$$\hat{M} = \frac{1}{T}\sum_{t=1}^{T} M(t) \quad . \tag{3.3}$$

An exact algorithm, that samples DAGs from modular distributions, has been implemented by Talvitie et al. [28]. It randomly samples from the space of DAGs. This approach results in lack of correlation among samples, therefore, no bias is present when DAGs are sampled from multimodal posterior distributions (this problem is frequent in structure MCMC algorithms described later on in this chapter). The downside of this method consists in computational limitations, when the algorithm is used to sample DAGs from networks where the number of nodes $N > 20$, because of exponential time and space complexities. A detailed description of the algorithm can be found in Section 5.3.1, since it has been used to generate reference distributions, in the experiments, for some specific networks. A different class of algorithms has to be used to estimate the posterior distribution of DAGs, when the number of variables of the network N increases.

3.1 Structure MCMC and MC^3 move

Structure MCMC is a popular algorithm that is used for Bayesian structure discovery. It avoids complications caused by the large number of possible states $|\mathcal{M}|$. It relies on performing specific moves to propose new DAGs, which are accepted according to a Metropolis–Hastings acceptance ratio.

3.1.1 Metropolis–Hastings algorithm

The Metropolis–Hastings algorithm (MH) [4, 5] is a MCMC method that is frequently used to sample from complex target distributions. In our case, the target distribution is the posterior distribution of DAGs. The output of the algorithm consists of a list of different states of the parameter (in our case M). The sampling process follows different steps. At first a initial state M_0 and a proposal distribution $q(M'|M)$ are defined, and an empty list that will contain the states of the parameter is created. Then, iteratively, a proposal for the next state of the chain is generated from the current state by sampling from $q(M'|M)$. An acceptance ratio for the new state is then computed.

$$a(M', M) = \min\left(1\ ,\ \frac{q(M|M')\ p(M'|D)}{q(M'|M)|\ p(M|D)}\right) \quad . \tag{3.4}$$

If a random sample from the uniform distribution Uniform$(0, 1)$ has a lower value than

$a(M', M)$, the current state takes the value of the proposed state. The current state is then attached to the list of states. The process is repeated for an arbitrary number T of iterations until a correct convergence to the distribution is assumed.

3.1.2 Madigan and York MC^3

The first structure MCMC algorithm for sampling DAGs, called *Markov Chain Monte Carlo Model Composition* or MC^3 model, has been proposed by Madigan and York [3]. The advantages of this model derive directly from a reduction of the space from which a DAG is sampled. More precisely, starting from a DAG $M(t) \in \mathcal{M}$ at iteration t in the Markov chain, a new DAG in the Markov chain $M'(t)$ is proposed if it is in the neighborhood of $M(t)$. Let us denote by $\text{nbd}(M)$ the neighborhood of the DAG M. In the Madigan and York model, the neighborhood of a DAG M is defined as the set containing the DAGs $M \in \mathcal{M}$ that have one edge more or one edge less than M. Hence, the transition matrix q has entries $q(M, M') = 0$ if $M' \notin \text{nbd}(M)$ and $q(M, M')$ is a constant if $M' \in \text{nbd}(M)$. The definition of the transition matrix q explains why the MC^3 model is more suitable for structure discovery, when the number of nodes of the network is large compared to an algorithm that samples randomly from the space of DAGs. In the MC^3 step, M' is sampled from a small set of DAGs leaving out most of the non-relevant low-scoring DAGs, which indeed would be taken into account when the previously described DAGs sampler described by Talvitie et al. [28] is used. However, high-scoring DAGs that are not in $\text{nbd}(M)$ are not considered.

A Metropolis–Hastings acceptance ratio is computed to assess how likely the proposed DAG is accepted [3]. We have

$$a(M', M) = \min\left(1\ ,\ \frac{|\text{nbd}(M)|\ p(M'|D)}{|\text{nbd}(M')|\ p(M|D)}\right)\ , \tag{3.5}$$

where M is the current state, M' is the proposed state and $|\text{nbd}(M)|$ represents the number of neighbors of M. The computation of the normalizing constant $p(D)$ can be omitted since it is equal for both terms $p(M'|D)$ and $p(M|D)$. Consequently, the posterior distribution can be replaced by the marginal likelihood. The marginal likelihood of the model is different from the BDeu score. The latter one is used as likelihood term of the model, therefore not uniform priors can also be used.

3.1.3 Giudici and Castelo MC^3

An improved version of the MC^3 move is described by Giudici and Castelo [6]. This model includes an additional possible move when transitioning from M to M'. The mentioned step is the reversal of an edge. In the Madigan and York model, the reversal of an edge could be performed in two distinct iterations: by first removing one edge, and then adding another edge with the opposite direction. Hence, by enabling a new move, the set of neighbors of any DAG $M(t) \in \mathcal{M}$ increases in size, improving the exploration efficiency of the structure MCMC runs, which usually are slow in mixing and exploring and result in high autocorrelation of the samples. This limitation is caused by the nature of the steps of the MC^3 algorithm. They are small, therefore, the current state M of the MCMC and the proposal M' differ by only one edge. Let us now describe in detail the three different steps of the improved version of the MC^3 move.

- **Addition**. An edge from X_i that points to X_j is formed if it does not create a cycle in the graph.
- **Removal**. The edge connecting X_i and X_j is removed. This step never causes the formation of a cycle in the graph.
- **Reversal**. It is a composition of the two steps mentioned above. The edge connecting X_i and X_j is removed and then added with the opposite orientation. This step, similarly to the addition step, is not always allowed because it can create a cycle in the graph.

Kahn's algorithm [29] provides an efficient method for checking for cycles in directed graphs, with a time complexity of $O(|V| + |E|)$.

3.1.4 Lazy MC^3

The Metropolis–Hastings ratio (Equation 3.5) states that all neighbors of the DAGs M and M' have to be found to obtain the cardinality of the neighbors of M and M'. This step can be computationally challenging. An alternative implementation, defined as Lazy MC^3 [30], which involves rejection sampling, allows us to ignore the neighborhood sizes. We achieve this by sampling M' from the space containing all directed graphs (not necessarily acyclic) that can be obtained by adding, removing, or reversing an edge starting from M. Furthermore, when M' is proposed, the maximum number of parents is not taken into account. If a graph that contains a cycle is sampled or if the sampled graph contains a node that has more parents than the

maximum allowed parents, then the proposal is rejected.

Allowing the proposal of graphs that include cycles and DAGs that exceed the number of maximum in-degree per node may seem counter-intuitive. However, M and M' have the same number of neighboring graphs (not necessarily acyclic) that are obtained by adding, removing, or reversing an edge. Since, the cardinality of the set of neighboring graphs of any DAG is equal to $N(N-1)$, the cardinality of the neighborhoods of M' and M is not needed for the computation of the Metropolis–Hastings ratio. This results in a lower computational cost of the move, compared to the Giudici and Castelo MC^3. The formula of the acceptance ratio of the Lazy MC^3 step is

$$a(M', M) = \min\left(1\ ,\ \frac{p(M'|D)}{p(M|D)}\right)\ . \tag{3.6}$$

The Metropolis–Hastings ratio now depends only on the scores of M and M'. Algorithm 1 shows in detail the Lazy MC^3 step. The proof of the convergence of the Lazy MC^3 sampler to the correct stationary distribution can be found in Appendix A.

Algorithm 1 Lazy MC^3

1: Draw u, v with $v \neq u$ at random from $E(M)$
2: $M' \leftarrow M$
3: **if** (u, v) is in $E(M')$ **then**
4: remove (u, v) from M'
5: **else if** (v, u) is in $E(M')$ & after removing (v, u) and adding (u, v) in M' no cycle in M' would be created & $|\pi_v| <$ max parents **then**
6: reverse (v, u) in M'
7: **else if** neither (u, v) nor (v, u) is in $E(M')$ & adding (u, v) would not create a cycle & $|\pi_v| <$ max parents **then**
8: add (u, v) in M'
9: **end if**
10: compute $a(M', M)$ ▷ Equation 3.6
11: $A \sim$ Bernoulli$(a(M', M))$
12: **if** $A = 1$ **then**
13: $M \leftarrow M'$

3.2 New Edge Reversal move

MC^3 move implies that the proposed DAG of a structure MCMC differs from the current DAG by one single edge. Therefore, convergence to the posterior distribution is slow. Additionally, a sampler that relies only on MC^3 moves can easily get stuck near local maxima, especially when the likelihood term is strong. This wrong behavior is described in detail in Section 5.4.1.

The *New Edge Reversal* (or REV) move was introduced by Grzegorczyk and Husmeier [7] to allow wider moves in the MCMC (M' does not differ from M by only one edge). REV move is computationally more demanding than MC^3. The basic concept of the REV move centers around the reversal of an edge. The reversed edge is selected by randomly sampling from a uniform probability distribution. In many cases, when the reversal move of MC^3 is applied to a DAG, cycles may be formed. When the REV move is performed, the parents set of the nodes at the end of the sampled edges are deleted, and new parents sets are sampled for both nodes. No cycles have to be formed when the parents sets are chosen and the max parents size for each node has to be taken into account. The whole process leads to a proposed DAG M', which may differ from the starting DAG M by many edges. The parents sets are sampled according to their score and the acceptance probability of the proposed DAG increases considerably. The proposed DAG has, most of the time, a higher score than the current DAG until the main score mode is reached.

To compute the Metropolis–Hastings ratio that assesses how likely M' is accepted, we need to define the probability of reaching M from M' (inverse move). Since the DAG M can be reached from M' in a univocal way, The Metropolis–Hasting ratio becomes [7]

$$a(M', M) = \min\left(1, \frac{H^{\dagger} \cdot Z^*(X_i|M_{\odot}, X_j) \cdot Z^*(X_j|M_{\oplus})}{H'^{\dagger} \cdot Z^*(X_i|M'_{\odot}, X_j) \cdot Z^*(X_j|M'_{\oplus})}\right) \quad , \tag{3.7}$$

where $H^{\dagger}$ and $H'^{\dagger}$ are the number of edges in M and M'. The notation $Z^*(X_n|M, X_m)$ represents a sum of local scores over all those parent sets π_i of X_n that contain X_m and do not create a cycle when inserted in the DAG M. $M_{\oplus}$ and $M_{\odot}$ are intermediate DAGs that are obtained by orphaning and adding new parents sets to the original DAGs, while $M'_{\oplus}$ and $M'_{\odot}$ are intermediate DAGs that are obtained by changing the proposed DAG M', to reach M (inverse step). More details on those intermediate DAGs can be found in the original paper [7].

3.3 Markov Blanket Resampling move

The *Markov Blanket Resampling* (MBR) move, introduced by Su and Borsuk [8], relies on proposing new DAGs for the MCMC, by changing the Markov blanket of a node. Similarly to the REV move, the proposed DAG M' may differ by many edges from the starting DAG M. Hence, the MBR move can help the structure MCMC to escape local maxima and explore the posterior distribution faster.

The change of the Markov blanket of a node is performed in several steps. Initially, a node X_i of the current DAG M is selected completely at random from a uniform probability distribution. Then X_i is orphaned and all the direct edges connecting the children of X_i and their parents are deleted (except the edges that connect X_i and their children). Subsequently, a new parents set is sampled for X_i and for all the children (the sampling order for the children is always randomized since a defined order would introduce bias in the results). The above-mentioned steps should not generate any cycle, and the number of max parents for each node should not be exceeded. The parents set are not proposed at random. Parents sets with higher score are proposed with higher probability. Hence, the value of the acceptance ratio of M' is not close to 0.

Another similarity to the REV move is that the inverse step is unique. The DAG M can be obtained by using a move of MBR from M' (inverse step) in a univocal way, and this property leads to an easy to compute acceptance ratio. The formula can be written as [8]

$$a(M', M) = \min\left(1, \frac{Z^*(X_i|M_0, \pi_i) \cdot \prod_{j=1}^{J} Z(X_i^j|M_j, X_i)}{Z^*(X_i|M_0', \pi_i') \cdot \prod_{j=1}^{J} Z(X_i^j|M_j', X_i)}\right) \quad , \tag{3.8}$$

where M_0 and M_0' refer to the starting DAG M and the proposed DAG M', after the sampled node X_i and its children have been orphaned. Each time a parents set is added, following the scheme described above, we obtain a new DAG M_j and the index j increases by one. Therefore, X_i^j refers to the j^{th} child of X_i.

3.4 Hybrid transition kernel

The structure MCMC that has been used in the experiments uses a mixture of MC^3 moves, MBR moves and REV moves. More precisely, in each iteration of the Markov chain, the probability of performing a REV move is p_{REV}, the probability of performing a MBR move is p_{MBR}, while the probability of performing a MC^3 move is

$p_{\mathrm{MC}^3} = 1 - p_{\mathrm{REV}} - p_{\mathrm{MBR}}$. Hence, the transition kernel K of this *hybrid* MCMC can be written as

$$K^{+}(M'|M) = p_{\mathrm{MC}^3} K^{\mathrm{MC}^3}(M'|M) + p_{\mathrm{REV}} K^{\mathrm{REV}}(M'|M) + p_{\mathrm{MBR}} K^{\mathrm{MBR}}(M'|M) \quad . \quad (3.9)$$

According to Tierney [31], since the transition kernel of the three moves have the same stationary distribution and K^{MC^3} is ergodic, also K^{+} converges to the same stationary distribution.

3.5 Considerations about structure MCMC

The experiments described in Section 5.4, show that the structure MCMC performs poorly when the number of data points is large and when the number of nodes N increases. The REV and MBR moves may allow an escape for the structure MCMC from a local mode, but they do not guarantee a correct convergence in all the circumstances. In Bayesian networks, the posterior distribution is usually multimodal, with sharp maxima separated by extended low-probability regions. Therefore, a simple structure MCMC may generate biased results. Parallel tempering is a beneficial tool that can solve this problem, leading to a correct exploration of the parameter space and to a correct convergence to the posterior distribution of DAGs.

4. Parallel tempering

In this chapter, the non-reversible parallel tempering (PT) scheme described by Syed et al. [10] is presented, with some implementations that are specifically added to increase the performance of the algorithm in the BN setting. PT involves the implementation of several MCMC algorithms that run in parallel and interact with each other. They sample from distributions at "different temperatures", which we will refer to as $U_c(M|D)$ for practical reasons. More in detail,

$$U_c(M|D) = \frac{p(M|D)^{\beta_c} p(M)^{1-\beta_c}}{Z(\beta_c)} \propto p(D|M)^{\beta_c} p(M) \quad , \tag{4.1}$$

where the partition $\mathcal{P} = \{\beta_0, ..., \beta_C\}$ of [0,1] is called annealing schedule and $Z(\beta_c)$ is the normalizing constant. A total number of $C+1$ chains are implemented and $\beta_0 = 0$, while $\beta_C = 1$. Therefore, $U_0(M|D) = U_0(M)$ is the prior distribution and $U_C(M|D)$ is the posterior distribution.

Parallel tempering was first introduced by Geyer [9]. Proofs regarding the correct convergence of the $C+1$ chains to the stationary distribution are shown in the paper. In the BN setting, the stationary distribution is given by

$$p(\boldsymbol{M}|D) = \prod_{c=0}^{C} U_c(M_c|D) \quad , \tag{4.2}$$

where $\boldsymbol{M}$ is a $C+1$ dimensional vector that contains the current states (DAGs) M_c of each chain.

A parallel tempering iteration is performed in two steps [10]. In the first step, called local exploration, each chain updates its current state according to the exploration kernel $K^{\text{expl.}}(M_c|c)$, where c is the index of the chain. In practice, each chain samples approximately from the distribution $U_c(M|D)$, associated to the temperature β_c. The second step consists in the communication phase, where swaps of states M_c are proposed among chains.

The posterior distribution $p(M|D) = U_C(M|D)$ is approximated by T samples where T is the total number of iterations performed. To identify each DAG sampled by the MCMC, let us use the notation $\boldsymbol{M}(t)$ and $M_c(t)$. We denote by $M_c(t)$ the DAG sampled by the chain at temperature β_c at time step t. To describe the current DAGs of the algorithm (last state of the different chains), the notation $\boldsymbol{M}$ and M_c without the time index t is used. Since the proposed states $\boldsymbol{M'}$ and M'_c of the MCMC depend only on the current state, the time index t is omitted when a single exploration and a single communication step are described. The different parallel tempering algorithms, presented in Section 4.2, share the same $K^{\text{expl.}}$, but they differ in their communication kernel $K^{\text{comm.}}$.

4.1 Local exploration

During the local exploration phase, each chain of the PT structure MCMC samples DAGs according to the hybrid transition kernel described in Section 3.4. As already mentioned in Section 2.1, the choice of the steps depends on the nature of the distribution. In general, the exploration phase can be performed using Metropolis–Hastings steps, Hamiltonian Monte Carlo steps, or Gibbs sampling steps. In Chapter 3 we already showed that the structure MCMC uses Metropolis–Hastings exploration steps.

When parallel tempering is enabled, the MH acceptance ratio of the exploration algorithm depends on the temperature β_c associated with the chain. We have $K_c^{\text{expl.}}(\cdot) = K_c^{+}(\cdot)$. Consequently, some adjustments have to be made to the acceptance ratio formulas of the MC^3 step (Equation 3.6), of the REV step (Equation 3.7) and of the MBR step (Equation 3.8). In the parallel tempering setting, let us denote the moves described in the previous chapter by MC^3_c, REV$_c$, and MBR$_c$, to take into account the different temperatures β_c, associated with the $C+1$ chains.

4.1.1 MC^3 move in PT structure MCMC

Equation 3.6 is modified to obtain the acceptance ratio of the MC^3_c step

$$a_c^{\text{expl.}}(M'_c, M_c) = \min\left(1\ ,\ \frac{U_c(M'_c|D)}{U_c(M_c|D)}\right)\ . \tag{4.3}$$

Applying the equivalence in Equation 4.1 we obtain

$$a_c^{\text{expl.}}(M'_c, M_c) = \min\left(1\ ,\ \frac{p(D|M'_c)^{\beta_c} p(M'_c)}{p(D|M_c)^{\beta_c} p(M_c)}\right)\ . \tag{4.4}$$

The log marginal likelihood is expressed using BDeu metric described in Section 2.5. Theorem 3 implies that the total score of any DAG is expressed as a sum of local scores. Therefore, we have

$$a_c^{\text{expl.}}(M'_c, M_c) = \exp\left(\min\left\{ 0 \ , \ \beta_c \sum_{i=1}^{N} \text{BDeu}[D|X_i, \pi'_{(i,c)}] + \ln(p(M'_c)) + \right.\right. \\ \left.\left. - \beta_c \sum_{i=1}^{N} \text{BDeu}[D|X_i, \pi_{(i,c)}] - \ln(p(M_c)) \right\}\right) \ , \tag{4.5}$$

where c refers to the index of the chain, β_c to its respective temperature and i represents the index of the node of the network (N total nodes). Therefore, $\pi_{(i,c)}$ represents the parents set of the node X_i in the DAG M_c, and $\pi'_{(i,c)}$ represents the parents set of the node X_i in the DAG M'_c.

4.1.2 REV and MBR moves in PT structure MCMC

A similar approach is used to modify the acceptance ratio of REV and MBR moves. The $Z^*(\cdot)$ terms in Equation 3.7 and Equation 3.8 contain a sum of local scores over all the parents set of a node that satisfy certain conditions. It has to be modified, to take into account the temperature β_c of each chain. Let us analyze, for example, a general term $Z_c^*(X_i|M_c^{\oplus})$ of the REV$_c$ move, for the chain c, corresponding to a temperature β_c. It can be expressed by

$$Z_c^*(X_i|M_c^{\oplus}) = \sum_{p=1}^{P} \exp\left(\text{BDeu}[D|X_i, \pi^{\oplus}_{(i,c,p)}] \cdot \beta_c + \ln(p(X_i|\pi^{\oplus}_{(i,c,p)})\right) \ , \tag{4.6}$$

where each index p refers to one of the P parents set of X_i that would not create a cycle when added to the DAG $M_c^{\oplus}$. Hence, $\pi^{\oplus}_{(i,c,p)}$ represents the p-th parents set added to the node X_i of the DAG $M_c^{\oplus}$. The notation $M_c^{\oplus} \neq M_c$ is used, since the intermediate steps of the REV move involve orphaning nodes and adding new parents set, therefore, these two DAGs are not the same. The term $p(X_i|\pi^{\oplus}_{(i,c,p)})$ refers to the prior probability of the single node X_i, when the parents set $\pi^{\oplus}_{(i,c,p)}$ is assigned to the node X_i. The use of the *log-sum-exp* trick is recommended not to encounter overflows in the computation. Algorithm 2 shows the pseudocode for the local exploration step. The variable Score is a dictionary that contains all local scores of each node, conditioned on the parents set.

Algorithm 2 Local exploration step

procedure LOCAL EXPLORATION STEP($\boldsymbol{M}$, p_{REV}, p_{MBR}, $\mathcal{P}_C$, Score, Prior)
  $R \sim$ Uniform (0,1)
  if $R < p_{\text{REV}}$ **then**
    for c in $\{0, 1, \ldots, C\}$ **do**
      $M_c \leftarrow \text{REV}_c$ move from M_c $(\boldsymbol{M}, \text{Score}, \text{Prior}, \beta_c)$
    end for
  else if $R < p_{\text{REV}} + p_{\text{MBR}}$ **then**
    for c in $\{0, 1, \ldots, C\}$ **do**
      $M_c \leftarrow \text{MBR}_c$ move from M_c $(\boldsymbol{M}, \text{Score}, \text{Prior}, \beta_c)$
    end for
  else
    for c in $\{0, 1, \ldots, C\}$ **do**
      $M_c \leftarrow \text{MC}^3_c$ move from M_c $(\boldsymbol{M}, \text{Score}, \text{Prior}, \beta_c)$
    end for
  end if

4.2 Communication kernels

Communication kernels define how the chains interact with each other and how swaps between states are proposed in the PT MCMC. In the BN setting, a swap can be performed by introducing a Metropolis–Hastings step among the different current DAGs that belong to $\boldsymbol{M}$. Communication is allowed among chains that have index that differ by one. More precisely, the proposed DAG for the state M_c, at temperature β_c, is the DAG M_{c+1}. It represents the current DAG of the PT MCMC at temperature β_{c+1}. Therefore, if a swap among the states M_c and M_{c+1} is accepted, we have [10]

$$\boldsymbol{M}^{(c,c+1)} = (M_0, M_1, \ldots, M_{c-1}, M_{c+1}, M_c, M_{c+2}, \ldots, M_C) \ . \tag{4.7}$$

The Metropolis–Hastings acceptance ratio among chains is obtained by

$$\begin{aligned} a^{(\beta_c, \beta_{c+1})}(\boldsymbol{M}) &= \min \left\{ 1 \ , \ \frac{p(\boldsymbol{M}^{(c,c+1)}|D)}{p(\boldsymbol{M}|D)} \right\} \\ &= \exp \left(\min \left\{ 0 \ , \ (\beta_{c+1} - \beta_c)\Big(V(M_{c+1}) - V(M_c)\Big) \right\} \right) \ , \end{aligned} \tag{4.8}$$

where

$$V(M_c) = -\ln \Big(\ p(M_c|D)/p(M_c) \ \Big) = -\ln \Big(p(D|M_c) \Big) \ . \tag{4.9}$$

Expanding Equation 4.8 we get

$$a^{(\beta_c,\beta_{c+1})}(\boldsymbol{M}) = \exp\left(\min\left\{0\ ,\ (\beta_{c+1}-\beta_c)\Big(\ln\big(p(D|M_c)\big) - \ln\big(p(D|M_{c+1})\big)\Big)\right\}\right). \tag{4.10}$$

The PT algorithms that will be tested and compared are the *deterministic even-odd* (DEO) swap algorithm (Section 4.2.2), the *stochastic even-odd* (SEO) swap algorithm (Section 4.2.1), and a parallel tempering scheme where the swap among chains with index c and $c+1$ is proposed randomly, in each communication step, from only one chain. For practical reasons, the latter sampler is defined as the *single random swap* (SRS) scheme. Details regarding this latter algorithm can be found below (Algorithm 3).

The performance of PT algorithms can be measured by assessing how well the "colder" chains (with temperature β_c close to 0) and the "warmer" chains (with temperature β_c close to 1) communicate. A good communication implies a good exploration of the parameter space because colder chains sample from distribution that do not present low-probability regions among modes. Therefore, the colder chains cannot get stuck near local modes. A mathematical quantification of this concept can be expressed by the *round trip rate* τ that is presented in Section 4.3.1. In the pseudocode, the vector $\boldsymbol{r}$, containing all rejection ratios between chains, is constantly updated, to obtain a good approximation of the *communication barrier* Λ described in Section 4.3.2.

Algorithm 3 SRS communication step

procedure SRS STEP($\boldsymbol{M}$, $\mathcal{P}_C$, $\boldsymbol{r}$, C, Score, Prior)
  c^* randomly sampled from the set $\{0, 1, \ldots, C-1\}$
  for c in $\{0, 1, \ldots, C-1\}$ **do**
    $a_c \leftarrow a^{(\beta_c,\beta_{c+1})}(\boldsymbol{M}, \text{Score}, \text{Prior}, \mathcal{P}_C)$ ▷ Equation 4.10
    $r_c \leftarrow r_c + 1 - a_c$
    if $c = c^*$ **then**
      $A \sim \text{Bernoulli}(a_c)$
      if $A = 1$ **then**
        swap M_c and M_{c+1} in $\boldsymbol{M}$
      end if
    end if
  end for

4.2.1 Stochastic even-odd scheme

In the SEO communication scheme [10], the indexes of the chains are stored in two different sets (even and odd). In each communication step, the swap is proposed simultaneously from all chains that have even indexes (therefore, the proposed states M_{c+1} have odd indexes), or the swap is proposed simultaneously from all the chains that have odd indexes (the proposed states M_{c+1} have even indexes). The choice of even or odd moves is obtained by randomly sampling from a Bernoulli(0.5) distribution. For example, in each communication step, if the outcome of sampling from the above-mentioned Bernoulli distribution is 0, swaps are proposed only from all the states with even index and vice versa if the outcome is 1. Algorithm 4 shows in detail a single SEO communication step.

Algorithm 4 SEO communication step

procedure SEO STEP($\boldsymbol{M}$, $\mathcal{P}_C$, $\boldsymbol{r}$, C, Score, Prior)
  $Z \sim$ Bernoulli(0.5)
  for c in $\{0, 1, \ldots, C-1\}$ **do**
    $a_c \leftarrow a^{(\beta_c, \beta_{c+1})}(\boldsymbol{M}, \text{Score}, \text{Prior}, \mathcal{P}_C)$ ▷ Equation 4.10
    $r_c \leftarrow r_c + 1 - a_c$
    if ($Z = 0$ and c is even) **or** ($Z = 1$ and c is odd) **then**
      $A \sim$ Bernoulli(a_c)
      if $A = 1$ **then**
        swap M_c and M_{c+1} in $\boldsymbol{M}$
      end if
    end if
  end for

4.2.2 Deterministic even-odd scheme

The DEO communication scheme has been proposed by Okabe et al. [11]. It works similarly to the SEO communication scheme. The swap is proposed only from even or odd chains, but even and odd indexes are not chosen by randomly sampling from a Bernoulli(0.5) distribution. Even and odd steps are fixed deterministically depending on the iteration (or time index t) of the PT MCMC. Hence, when t is even, the swaps are proposed only from the states with even indexes, while, when t is odd, the swaps are proposed only from the current states of odd chains. The DEO algorithm is non-reversible since the Markov chain cannot backtrack its movements and does not satisfy the properties of random walks. Reversibility is seen as an inefficient property

of Markov chains, and non-reversibility leads to a better exploration of the parameter space [32]. Indeed, after the chain with index c swaps its state with the chain associated with index $c+1$, in the next step, the state cannot go back to the chain at index c, but it can only stay at index $c+1$ or reach $c+2$. More mathematical details on the theoretical performance of the different schemes are presented in Section 4.4. The pseudocode for the DEO communication scheme can be found below [10].

Algorithm 5 DEO communication step

procedure DEO STEP($\boldsymbol{M}$, $\mathcal{P}_C$, $\boldsymbol{r}$, C, t, Score, Prior)
  for c in $\{0, 1, \ldots, C-1\}$ **do**
    $a_c \leftarrow a^{(\beta_c, \beta_{c+1})}(\boldsymbol{M}, \text{Score}, \text{Prior}, \mathcal{P}_C)$ ▷ Equation 4.10
    $r_c \leftarrow r_c + 1 - a_c$
    if (t is even and c is even) **or** (t is odd and c is odd) **then**
      $A \sim \text{Bernoulli}(a_c)$
      if $A = 1$ **then**
        swap M_c and M_{c+1} in $\boldsymbol{M}$
      end if
    end if
  end for

4.3 Important quantities for PT analysis

In the previous section, it is mentioned that good communication among all chains is needed, to sample efficiently from the posterior distribution. In "colder" chains, the likelihood term lightly influences the distribution, and sampling from a distribution that more closely resembles the prior distribution allows a better exploration of the parameter space.

To optimize the computation, Syed et al. [10] suggest to run in parallel $C+1$ machines, that sample from $C+1$ temperatures, using a different core for each machine. Each machine, at iteration t, is associated with a current DAG and a temperature. From an algorithmic point of view, there are two different ways to implement a PT MCMC. In the first implementation, every machine is associated to the same temperature during the whole sampling process, therefore, when a swap among chains is accepted, the current DAGs of the respective machines are swapped. Alternatively, when a swap is accepted, the indexes of the temperatures of the machines are swapped, instead of the current DAGs.

4.3.1 Round trip rate

The *round trip rate* τ is widely used as a measure of efficiency for parallel tempering schemes. A round trip occurs, for example, in a machine j at temperature $c = 0$, when the index associated with the temperature of the chain goes from 0 to C and back to 0. More details can be found in the original paper [10]. $\mathcal{T}_k^j$ represents the number of iterations t needed by the machine j to perform the k-th round trip. Defining as $\mathcal{T}$, the random variable associated to $\mathcal{T}_k^j$, we have [10]

$$\tau = \frac{C+1}{\mathbb{E}[\mathcal{T}]} \quad . \tag{4.11}$$

4.3.2 Global and local communication barrier

The global and local communication barrier are quantities that describe the relations among the temperatures of the chains and their rejection ratios. Therefore, a good estimation of the communication barrier is the basis for the optimization process of PT MCMC algorithms.

Theoretical definition of the communication barrier

Let us define the acceptance ratio $s(\cdot)$ and the rejection ratio $r(\cdot)$ as functions of the temperatures. Those functions output a probability, therefore, their codomain is $[0, 1]$. Next, we give a brief overview of the approach of Syed et al. [10]. Starting from Equation 4.10

$$\begin{aligned} s(\beta, \beta') &= \mathbb{E}[a^{(\beta,\beta')}(M_\beta, M_{\beta'})] \\ &= \mathbb{E}\left[\exp\left(\min\left\{ 0,\ (\beta' - \beta)\left(\ln\Big(p(D|M_\beta)\Big) - \ln\Big(p(D|M_{\beta'})\Big)\right)\right\}\right)\right] \quad , \end{aligned} \tag{4.12}$$

where M_β and $M_{\beta'}$ are the current DAGs of a PT MCMC at temperatures β and β'. The rejection ratio among two temperatures is obtained by computing

$$r(\beta, \beta') = 1 - s(\beta, \beta') \quad . \tag{4.13}$$

The incremental ratio $\lambda(\beta)$, at temperature β, is called instantaneous rate of rejection [10]. The formula for the incremental ratio is given by

$$\lambda(\beta) = \lim_{\delta \to \inf} \frac{r(\beta, \beta + \delta)}{|\delta|} \quad . \tag{4.14}$$

The communication barrier is defined as

$$\Lambda(\beta) = \int_0^{\beta} \lambda(\beta')\, d\beta' \quad . \tag{4.15}$$

$\Lambda(1) = \Lambda$ is called *global* communication barrier and $\Lambda(\cdot)$ function is called *local* communication barrier. The latter one is a monotone increasing function, and its codomain is always ≥ 0. Predescu and Ciobanu [33] show that the function $\Lambda(\cdot)$ is needed to calculate the rejection ratio between chains at different temperatures

$$r(\beta, \beta') = |\Lambda(\beta') - \Lambda(\beta)| + O(|\beta' - \beta|^3) \quad . \tag{4.16}$$

Therefore given any annealing schedule $\mathcal{P}_C$ we have [10]

$$\sum_{c=0}^{C-1} r_c = \Lambda + O(C||\mathcal{P}_{\bar{C}}||^3) \quad , \tag{4.17}$$

where $||\mathcal{P}_{\bar{C}}|| = \max_c |\beta_c - \beta_{c-1}|$ and $\mathbf{r}$ is not anymore a function, but it is C-dimensional vector, where each entry r_c represents the rejection ratio between chains c and $c+1$.

Empirical estimation of the communication barrier

Starting from Equation 4.17, and assuming $||\mathcal{P}_C|| \to 0$, we can get a discrete estimator $\hat{\Lambda}(\cdot)$ of the local communication barrier, at fixed β_k

$$\hat{\Lambda}(\beta_k) = \sum_{c=0}^{k-1} \hat{r}_c \quad , \quad \text{where} \quad \hat{r}_c = \frac{1}{T} \sum_{t=1}^{T} \left(1 - a^{(\beta_c, \beta_{c+1})}\right) \quad , \tag{4.18}$$

and t refers to the index of the PT scan. The global communication barrier is computed by setting $k = C$. We note that as $C \to \infty$, the norm $||\mathcal{P}_C|| \to 0$, therefore, a large starting number of chains C must be chosen to obtain a reasonable approximation.

Equation 4.18 gives a discrete approximation of the local communication barrier. To obtain a continuous function $\bar{\Lambda}$, monotone cubic spline interpolation, described by Fritsch and Carlson [34], is performed. Algorithm 6, provided by Syed et al. [10], describes the steps needed for the computation of $\bar{\Lambda}(\cdot)$.

4.4 Comparing DEO, SEO, and SRS schemes

The theoretical value of the expected value of the iterations that are needed for a machine to perform a round trip can be computed to optimize τ. This measure was introduced by Nadler and Hansmann [35] for reversible algorithms. The results for the DEO and SEO schemes have already been shown by Syed et al. [10]. The steps that

Algorithm 6 Computing $\bar{\Lambda}(\cdot)$

procedure LOCAL COMMUNICATION BARRIER($\mathcal{P}_C$, $\hat{\boldsymbol{r}}$, C)
  for c in $\{0, 1, \ldots, C\}$ **do**
    Compute $\hat{\Lambda}(\beta_c)$ ▷ Equation 4.18
  end for
  $S \leftarrow \{(\beta_0, \hat{\Lambda}(\beta_0)), (\beta_1, \hat{\Lambda}(\beta_1)), \ldots, (\beta_C, \hat{\Lambda}(\beta_C))\}$
  compute monotone cubic spline $\bar{\Lambda}(\cdot)$ using the set of points S
  return $\bar{\Lambda}(\cdot)$

lead to Equation 4.21 for the SRS scheme are shown in detail in Appendix B. Starting from

$$\mathbb{E}_{\text{SEO}}[\mathcal{T}] = 2(C+1)C + 2(C+1)\sum_{c=0}^{C-1} r_c/s_c \quad , \tag{4.19}$$

$$\mathbb{E}_{\text{DEO}}[\mathcal{T}] = 2(C+1) + 2(C+1)\sum_{c=0}^{C-1} r_c/s_c \quad , \tag{4.20}$$

$$\mathbb{E}_{\text{SRS}}[\mathcal{T}] = C^2(C+1) + C(C+1)\sum_{c=0}^{C-1} r_c/s_c \quad , \tag{4.21}$$

where r_c represents the rejection ratio among chain c and $c+1$, and s_c is the acceptance ratio ($s_c = 1 - r_c$). By integrating Equations 4.19, 4.20, 4.21, and Equation 4.11, we see that the DEO scheme performs better compared to the other communication schemes. Additionally, Syed et al. [10] show that, when $||\mathcal{P}_C|| \to 0$,

$$\Lambda \leq \sum_{c=0}^{C-1} r_c/s_c = \Lambda + O(||\mathcal{P}_C||) \quad . \tag{4.22}$$

Therefore, the asymptotic round trip rate τ becomes

$$\tau_{\text{SEO}}(\mathcal{P}_C) \approx \frac{1}{2C + 2\Lambda} \to 0 \quad , \tag{4.23}$$

$$\tau_{\text{DEO}}(\mathcal{P}_C) \approx \frac{1}{2 + 2\Lambda} > 0 \quad , \tag{4.24}$$

$$\tau_{\text{SRS}}(\mathcal{P}_C) \approx \frac{1}{C^2 + C\Lambda} \to 0 \quad . \tag{4.25}$$

The SRS aglorithm, as expected, performs worse compared to the DEO and SEO algorithms. We also see that the SEO communication scheme, performs poorly when $C \to \infty$ (consequence of $||\mathcal{P}_C|| \to 0$), since the colder and warmer chains stop

communicating and the round trip rate τ drops to 0. This problem is bypassed when the DEO communication kernel is used since τ approaches a constant when $C \to \infty$ [10].

A practical approach may be useful for understanding the reasons behind the above-shown results. By setting $C \to \infty$, the acceptance ratio between communicating chains approaches 1 (Equation 4.16) since the absolute value of the difference of the temperatures of communicating chains approaches 0. Using a deterministic scheme, it is easier for any machine j to reach one of the two boundaries represented by the chains at temperature β_0 or β_C, since the swaps are proposed to reduce the change in direction of the swaps. Let us assume that all acceptance rates for communicating chains are $s^* = 0.9999$. Using the DEO scheme, any machine j, performs a swap from a chain at temperature β_c to a chain at temperature β_{c+20} in 20 steps with probability equal to $(s^*)^{20} \approx 0.998$. Using the SEO scheme, the probability becomes $(0.5 \cdot s^*)^{20} \approx 0$ since a change in direction of the swap is proposed, with probability equal to 0.5, after each iteration. Therefore, non-reversibility is seen as a positive trait and well-tuned non-reversible communication schemes outperform reversible communication schemes.

4.5 Novel tuning routine for PT algorithms

The performance of the PT algorithms is strongly affected by the choice of hyperparameters. Not optimal values for the annealing schedule $\mathcal{P}_C$ and for the number of chains $C+1$ would result in a lack of communication between some contiguous chains, leading to incorrect exploration of the parameter space. Hence, hyperparameter tuning has to be performed, to maximize the efficiency of the algorithm and to avoid potential problems. Temperature tuning is not a new concept. Kone and Kofke [36] and Atchadè et al. [12] suggested a tuning routine for the annealing schedule, which has been applied for reversible PT schemes.

Hyperparameter tuning relies on τ optimization [32, 37] . Therefore, Equations 4.19, 4.20, and 4.21, show that the quantity $\sum_{c=0}^{C-1} r_c/s_c$ has to be minimized to find an optimal value of the round trip rate τ [10]. Applying the constraints $\sum_{c=0}^{C-1} r_c = \Lambda$ and $r_c > 0$ for all c, using Lagrange multipliers, we get a solution where the optimal rejection ratio between chains is constant [12, 32, 33]. In mathematical notation $r_c = r^*$ for all c.

To optimize the round trip rate τ, the annealing schedule has to be built keeping the rejection ratio among chains constant, therefore [10],

$$\Lambda(\beta_c^*) = \frac{c\Lambda}{C} \quad , \tag{4.26}$$

where $\Lambda(\cdot)$ is a function. Starting from Equation 4.26, the optimal values for β_c^* can be computed by applying the bisection formula [10, 38]. This method is used to find a root for continuous functions. It consists in iteratively dividing in two an interval and selecting the sub-interval that contains a change of sign. The process of finding optimal temperatures is described in detail in Algorithm 7.

Equation 4.26 shows that an optimal estimation of the temperatures is based on a correct estimation of the local communication barrier $\bar{\Lambda}(\cdot)$ (Algorithm 6). This condition is achieved when the starting number of chains $C + 1 \to \infty$. Syed et al. [10] describe a tuning phase, in which each core of the GPU is used to run a chain associated with a specific temperature. We proposed a novel method to tune the above-mentioned parameters, that does not require GPU and does not require a huge number of chains that are implemented in parallel (popular NVIDIA GPUs may have many thousands of cores [39]). This new method relies on two tuning phases and requires a starting value, for the number of chains, that is significantly lower than the average number of cores of GPUs, speeding up the whole tuning process when GPU is not available.

Algorithm 7 Optimal $\mathcal{P}_C^*$

1: **procedure** OPTIMAL ANNEALING SCHEDULE ($\Lambda(\cdot)$, C)
2: $\Lambda \leftarrow \Lambda(1)$
3: **for** c in $\{0, 1, 2, \ldots, C\}$ **do**
4: Compute β_c^* that satisfies $\Lambda(\beta_c^*) = \frac{c\Lambda}{C}$ using bisection.
5: **end for**
6: **return** $\mathcal{P}_C^* = \{\beta_0^*, \beta_1^*, \beta_2^*, ..., \beta_C^*\}$

4.5.1 First tuning phase

The first tuning phase is performed to find the optimal number of chains for the PT MCMC. The optimal value of the rejection ratio between chains, reported by Syed et al [10], for DEO algorithm is $r^* = 0.5$. It is different from the optimal value for reversible communication schemes $r^* = 0.77$ that is frequently reported in the literature [12, 32, 36]. By substituting those values into equation $r^* = \Lambda/C$ we obtain the optimal estimates $\bar{C}_{\text{DEO}} \approx 2\Lambda$ and $\bar{C}_{\text{SEO}} = \bar{C}_{\text{SRS}} \approx \Lambda/0.77$.

Reducing the initial number of chains

Equation 4.16 shows that a large starting number of chains has to be used, to reduce the error in the estimation of $\bar{\Lambda}$. On the other hand, when GPU is not available, the choice of an extremely large value for the initial number of chains $C+1$ may result in a computational cost that is not feasible. A starting value $C+1 \simeq 4N$ where N is the number of nodes of the network, has been used for the experiments. This implementation leads to an accurate approximation of the global communication barrier $\bar{\Lambda}$. The local communication barrier function is not well approximated, but this limitation does not affect the result since only the global communication barrier value $\bar{\Lambda}$ is needed to compute an optimal estimation of the number of chains $\bar{C}+1$.

Several tuning steps

An estimator of the local communication barrier $\bar{\Lambda}(\cdot)$, as already expressed by Syed et al. [10], can be found by iteratively updating the annealing schedule. The main algorithm starts with an arbitrary annealing schedule that is updated after every tuning step. The tuning steps are performed several times, each time involving a different number of iterations t. The number of tuning steps of the first tuning phases is given by the relation $n_{\text{tune}} = \text{int}(\log_2 T_{\text{train}}) - 2$, where T_{train} is the number of total iterations that are reserved for the first tuning phase and $\text{int}(\cdot)$ is a function that rounds float numbers to the lowest integer. The number of iterations of each tuning step of the first tuning phase follows the formula 2^l where $l = \{1, 2, \ldots, n_{tune}\}$. Therefore, the number of iterations of each tuning step is $I_{\text{tune}} = \{2, 4, 8, \ldots, T_{\text{train}} - T_{\text{performed}}\}$ where $T_{\text{performed}}$ represents the sum of the iterations of each tuning step that has already been performed before the last tuning iteration. T_{tune} contains the time indexes t associated with the update of the annealing schedule. It is obtained by applying the cumulative sum on I_{tune}. Hence $T_{\text{tune}} = \{2, 6, 14, \ldots, T_{\text{train}}\}$.

A practical example may be useful to avoid confusion in the notation. Let us assume that $T_{\text{train}} = 5000$, therefore, 5000 iterations are used for the first tuning phase. Hence,

$$n_{\text{tune}} = \text{int}(\log_2(5000)) - 2 = 10 \quad ,$$

$$I_{\text{tune}} = \{2^1, 2^2, 2^3, \ldots, 2^{10}, 5000 - \textstyle\sum_{k=1}^{10} 2^k\} = \{2, 4, 8, \ldots, 1024, 5000 - 2046\} \quad ,$$

$$T_{\text{tune}} = \{2, 6, 16, \ldots, 2046, 5000\} \quad .$$

The training routine described above is suggested because the standard annealing schedule, selected at the beginning, is most of the time far from the optimal annealing schedule. Hence, few iterations for the first training phases are sufficient to obtain a better $\mathcal{P}_{\bar{C}}^*$. Each time $\mathcal{P}_{\bar{C}}^*$ is updated, more iterations are needed to obtain a more accurate estimate of the optimal annealing schedule.

After several tuning steps, the value of global communication barrier stabilizes around a fixed value. The experiments in Section 5.7.3 show that this value is a reliable estimator of $\bar{\Lambda}$, hence, it can be used to calculate the optimal number of chains $\bar{C}+1$ that minimizes the round trip rate τ.

4.5.2 Second tuning phase

Syed et al. [10] state that the tuning phase ends right after $\bar{C}$ is found. Using the same $\bar{\Lambda}(\cdot)$ calculated in the last tuning step, using bisection, $\bar{C}+1$ values for the temperatures β_c can be obtained. In practice, in the BN setting, running a PT algorithm that uses the temperatures β_c obtained after the first tuning phase described above, results in rejection ratios between chains that are not constant. The rejection rates between chains have high variance and some $r_c \approx 0$, while some $r_c \approx 1$. Since the experiments are performed using a strong likelihood term, the accuracy of the annealing schedule is extremely relevant. In sharp modes, DAGs have high score. Starting from Equation 4.12, we want the acceptance ratio among chains c and $c+1$ to be

$$\begin{aligned} s(\beta_c, \beta_{c+1}) &= \mathbb{E}\Bigg[\exp\Bigg(\min\Bigg\{0,\ (\beta_{c+1}-\beta_c)\Bigg(\ln\Big(p(D|M_{\beta_c})\Big) - \ln\Big(p(D|M_{\beta_{c+1}})\Big)\Bigg)\Bigg\}\Bigg)\Bigg] \\ &= 0.5 \quad . \end{aligned} \tag{4.27}$$

The term $\Big|\ln\Big(p(D|M_c)\Big) - \ln\Big(p(D|M_{c+1}\Big)\Big|$ may be extremely large, and it gets larger in module, when more data points are added. Therefore, the value $(\beta_{c+1}-\beta_c)$ should always be more accurate, the more the log likelihood term is strong. Even small biases in the computation of the annealing schedule can result in rejection rates among chains that are far from the optimal value of 0.5.

The high variance in the rejection rates r_c is also a consequence of the choice of the starting number of chains, since the error gets lower when $C+1$ increases (Equation 4.16). Hence, another tuning phase is recommended. In the second

tuning phase, the value of the number of chains $\bar{C} + 1$ remains fixed and only the annealing schedule $\mathcal{P}_C^*$ is optimized to obtain a constant rejection rate r^* between communicating chains. Similarly to the first tuning phase, the second tuning phase is divided into different steps, performed by iteratively updating the annealing schedule to obtain a constant rejection ratio r^* among chains. The second tuning phase does not significantly change the value of the global communication barrier. Indeed, after this tuning routine, the value of the rejection rate between communicating chains is $r_c \approx r^* \approx 0.5$ for every c.

The choice of the number of iterations t used during each tuning step of the second tuning phase might be different. The same idea of updating the annealing schedule every 2^l iterations can be used to reach a fixed value that is always more and more accurate. In the experiments, every tuning step is performed after a fixed amount of sampling iterations (I_2 in Algorithm 8), to obtain a sampler that updates the temperatures fast and, consequently, allows fast escapes from local modes. The annealing schedule is updated, in the second phase, for a fixed number of times (variable Rounds in Algorithm 8), when dynamic tuning is not used.

4.5.3 Dynamic tuning

Dynamic tuning is enabled by not stopping the second tuning phase. In practice, the number of tuning rounds, of the second tuning phase, is set to ∞. The idea behind this implementation is to develop a versatile sampler that performs effectively even when the hyperparameters are not correctly initialized. In addition, this tuning scheme is more versatile and always ensures an escape from local maxima since the annealing schedule is continuously updated to achieve optimal communication among chains. The Metropolis–Hastings ratio, of both local exploration and communication steps, is based only on the score of the DAGs. Many DAGs may have similar scores and, at the same time, the scores of their neighboring DAGs may vary consistently. Therefore, DAGs with similar scores may communicate differently with their neighbors. Dynamic tuning was introduced to overcome this problem and ensure a constant rejection ratio r_c between all the chains.

It is an open question whether dynamic tuning samplers generate unbiased posterior distributions, therefore, assumptions regarding the validity of this tuning routine relies only on empirical analysis and experimental results. In practice, by looking at the experiments shown in Sections 5.7 and 5.8, dynamic tuning does not change the annealing schedule significantly over time and it ensures a constant rejection ratio r_c

between all the chains.

4.6 PT structure MCMC algorithm

Algorithm 8 shows in detail the PT structure MCMC algorithm. It returns a list of DAGs, sampled according to the posterior distribution. The communication scheme has to be selected at the beginning and the different algorithms, described in the previous sections, are deployed in a precise order. Different structure priors for the DAGs can be chosen, hence, the formulas of the acceptance ratio of the local exploration step and of the communication step have to be changed accordingly.

T_{train} refers to the number of iterations performed during the first tuning phase, while $T_{\text{iter.}}$ refers to the number of iterations performed after the first tuning phase. If GPU is available and $\mathcal{C}$ cores are available, it is suggested to use $C+1=\mathcal{C}$ as the starting number of chains for the tuning phase [10]. If only one core is available, $C+1$ has to be chosen carefully, since a small value for $C+1$ would result in a poor estimate of $\bar{\Lambda}(\cdot)$ while a large value of $C+1$ would result in a high computational cost. In line 25 of Algorithm 8, the number of chains chosen after the first tuning phase is set to 2Λ, also for reversible algorithms. This approximation ensures a fair comparison of the round trip rate τ for all communication schemes.

Algorithm 8 Structure MCMC with parallel tempering

procedure PT STRUCTURE MCMC (N, max parents, T_{train}, $T_{\text{iter.}}$, p_{REV}, p_{MBR}, C, I_2, Rounds, Dynamic, Score, Prior)
  if Dynamic = True **then**
    Rounds $\leftarrow \infty$
  end if
  $\mathcal{P}_C \leftarrow$ default annealing schedule
  $\boldsymbol{M} \leftarrow$ generate $C+1$ random DAGs (with N nodes and specified max parents)
  COMMUNICATION STEP $\leftarrow$ select between {DEO, SEO, SRS}
  compute n_{tune}, I_{tune}, and T_{tune} ▷ Equations in Section 4.5.1
  $\boldsymbol{r} \leftarrow C$ dimensional vector with 0 entries
  $k \leftarrow 0$
  Post_DAGs $\leftarrow$ empty list ▷ posterior distribution of DAGs
  for t in $\{0, 1, \ldots, T_{\text{train}} + T_{\text{iter.}} - 1\}$ **do**
    $k \leftarrow k + 1$
    perform LOCAL EXPLORATION STEP ($\boldsymbol{M}$, p_{REV}, p_{MBR}, $\mathcal{P}_C$, Score, Prior)
    perform COMMUNICATION STEP ($\boldsymbol{M}$, $\mathcal{P}_C$, $\mathbf{r}$, C, t, Score, Prior) ▷ also $\boldsymbol{r}$ is updated
    append M_C to Post_DAGs
    $t_2 \leftarrow (t - T_{\text{train}})$ ▷ time index for the second tuning phase
    if (t in T_{tune}) or (t_2 is multiple of I_2 and $0 \leq t_2/I_2 \leq$ Rounds) **then**
      $\boldsymbol{r} \leftarrow \boldsymbol{r}/k$
      $k \leftarrow 0$
      $\Lambda(\cdot) \leftarrow$ LOCAL COMMUNICATION BARRIER ($\mathcal{P}_C$, $\boldsymbol{r}$, C)
      $\mathcal{P}_C \leftarrow$ OPTIMAL ANNEALING SCHEDULE ($\Lambda(\cdot)$, C)
    end if
    if $t = T_{\text{train}}$ **then** ▷ end of the first tuning phase
      $C \leftarrow 2\Lambda$ ▷ optimal number of chains $C+1$
      $\mathcal{P}_C \leftarrow$ OPTIMAL ANNEALING SCHEDULE ($\Lambda(\cdot)$, C)
    end if
  end for
  return Post_DAGs

5. Experiments

Several experiments were conducted to test the performance of different parallel tempering schemes and to observe the properties of the structure MCMC moves. The goal was to study the limitations of the above-mentioned model and to develop an efficient DAGs sampler. In the first set of experiments, the weaknesses of the structure MCMC, without any parallel tempering scheme, are underlined to illustrate how the structure MCMC can be improved. In the second set of experiments the advantages and limits of each move (MC^3, REV, and MBR) are shown and hyperparameter tuning has been performed to find the best values for the probability of performing each move, to obtain faster convergence.

In the third set of the experiments, the performance of the different parallel tempering schemes is compared. The DEO scheme is not exclusively compared to the SEO scheme and to the SRS scheme, but also to other DEO schemes where the annealing schedule is calculated using different methods. The performance of the DEO sampler with fixed standard temperature values is compared with the performance of other DEO samplers where the temperatures are tuned.

Additional experiments are performed to assess whether the DEO PT algorithm converges to the posterior distribution when the second tuning routine for the annealing schedule is not stopped after a fixed number of tuning steps (dynamic tuning). This is a pure experimental analysis and it relies on the idea that merging different MCMC runs that use different parallel tempering annealing schedules should generate a distribution that approximates correctly the posterior distribution. In the last section, the behavior of the temperatures, during the two tuning phases, is shown. The temperatures are plotted as functions of the tuning iterations, to see how dynamic tuning affects the sampling process.

5.1 Experimental setup

The *Python* programming language has been used to implement all the steps and the main algorithm. The package *NumPy* [40] in particular is used, due to its optimized processing capabilities, to represent DAGs as adjacency matrices. The adjacency matrix representation consists of matrices $A_{N\times N}$ where N is the number of variables. It has binary entries (0 or 1) and the value $A_{ij} = 1$ represents the presence of an edge going from X_i to X_j. The BDeu local scores have been calculated using PyGOBNILP [41]. PyGOBNILP's main objective focuses on score-based Bayesian network structure learning (BNSL) using integer programming (IP). The computations have been performed using a nested cluster for high computing performance developed by the University of Helsinki, called Turso. The algorithms have been implemented from scratch and the source code can be found in the GitHub repository *Dynamic DEO Structure MCMC for sampling DAGs* [†].

5.2 Synthetic and empirical datasets

Different datasets have been used for the experiments. Some of those are artificially generated from benchmark networks. In these cases, they are randomly generated from a network where the structure and the conditional probability tables for the whole DAG is known. Empirical datasets contain only observed data and no information regarding the generating DAG is given.

Benchmark networks

The ASIA dataset consists in 10 000 data points sampled from the Asia benchmark network [17], which has already been shown in Figure 2.1. This network has been widely analyzed in Bayesian structure learning tasks. It is characterized by 8 binary variables related to lung diseases and events (for example, exposure to X-rays, smoking, and visits to Asia). The maximum in-degree has been set to three by default. Another synthetic dataset (INS), that contains 5 000 data points, has been sampled from the Insurance benchmark network [42]. It contains 27 discrete variables (multinomial) and it shows the causal relations of variables that are important in the car insurance field. Also in this case the maximum in-degree has been set to three since the node with the highest number of parents, in the network, has three parents.

[†]The GitHub repository contains the code of the different structure MCMC algorithms, the datasets used in the experiments, and an example of DAGs sampling using the DEO structure MCMC. Link: https://github.com/danielone8/dyn-deo-dags.

The number of data points $|D|$ for ASIA and INS is large to test the algorithms in challenging situations; when $|D|$ is large, the prior tends to lose influence on the posterior and the likelihood term, which has more influence, generates low-probability regions in the posterior distribution.

Empirical datasets

Two empirical datasets, called Mushroom (MUSH) [43] and Zoo (ZOO) [44], have been used to compare the efficiency of the different algorithms. For those two datasets, there is no reference DAG, therefore the maximum in-degree used in the experiments can be arbitrary. The Mushroom dataset used in the experiments contains 4 000 data points randomly sampled from the original dataset. Not all data points have been used since the posterior distribution, obtained by using the complete dataset, would have generated a probability mass function that was bigger than 0 only for few DAGs. It is important to test the validity of the algorithms also in settings where the posterior distribution presents more extended high-probability regions, to see how fast the MCMC runs mix and converge. The Mushroom dataset contains features of 23 species of gilled mushrooms in the Agaricus and Lepiota family. The maximum in-degree has been set to four.

The last empirical dataset is Zoo dataset. It describes 17 features of 101 animals from a zoo. This dataset has been used to check the performance of the algorithms in a setting where the likelihood term is not predominant, therefore even simple samplers that do not have any parallel tempering scheme implemented could in theory sample correctly from the posterior distribution. The maximum-in degree in this case has been set to three.

5.3 Generation of reference distributions

The posterior distribution obtained by the different structure MCMC runs is compared to those obtained using the packages BIDA (Bayesian Intervention Distribution Adjustment) by Pensar et al. [45] and Modular DAG Sampling by Talvitie et al. [28]. For practical purposes, we will refer to the latter algorithm as MDS.

5.3.1 Modular DAG sampling

As already mentioned in Section 3, a reference distribution can be generated by sampling using MDS. The algorithm randomly samples from the space of DAGs. To

approximate the posterior distribution, a suitable number of DAGs is sampled for each network. The accuracy of the approximation does not depend on the shape of the posterior distribution, hence the algorithm can be applied safely to sample DAGs from multimodal distributions. In other words, the DAGs are sampled independently and they lack of autocorrelation. MDS samples from a modular distribution (the score is calculated by summing the log local score of each node), therefore, the score function of the DAGs is the same of the one used for the structure MCMC.

MDS is inspired by the recurrences of Tian and He [46]. Some modifications have been applied since the approach of Tian and He does not involve any direct sampling of DAGs. Hence, the root-layering approach of Kuipers and Moffa is used [47, 48]. This type of sampling process is divided into two different phases. At first, a partition of the nodes into layers is sampled. Subsequently, a DAG conditional on the root-layering is sampled. The algorithm requires $O(3^N)$ time for pre-processing and $O(2^N)$ time per sample.

5.3.2 Bayesian Intervention Distribution Adjustment

The APS tool for causal effect discovery using BIDA, can be found in the GitHub repository *BIDA* [†]. It does not involve sampling DAGs. It is built on top of the IDA algorithm [49, 50], which relies on finding a CPDAG from the data. As already expressed in the Introduction, especially when the number of available data points is small, discovering only the highest scoring DAG leads to a significant loss of information. BIDA performs Bayesian model averaging in order to find a distribution of the causal effects among variables. BIDA requires $O(3^N N)$ computed in time and $O(2^N N)$ computed in space to calculate the probabilities of the $N(N-1)$ ancestors relation probabilities and the $2^{N-1}N$ parents set probabilities. The maximum number of nodes N, which both BIDA and MDS algorithms can handle, is roughly $N = 20$.

5.3.3 Comparing BIDA and MDS on ASIA

Both BIDA and MDS algorithms can be applied safely to analyze the Asia benchmark network, since the number of nodes of the model is small (8 nodes). To generate the reference distribution using MDS, two million DAGs were sampled.

[†]The GitHub repository contains the code and the APS tool for the computation of parent set and ancestor relation posterior probabilities. Link: https://github.com/jopensar/BIDA.

Score analysis for MDS

In Figure 5.1 the two million samples are divided into 40 sets containing 50 000 DAGs each. The histogram of each set is compared with the others, to check whether they converge to the same distribution. All the histograms (each color represents one set

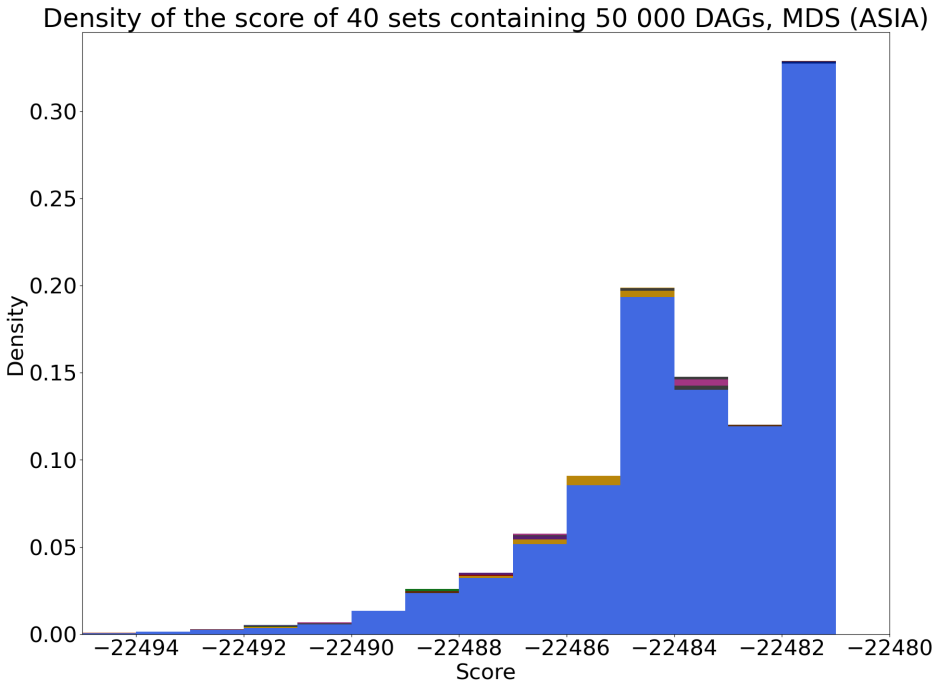

Figure 5.1: Histogram of the score of 40 MDS sets containing 50 000 samples. Each color is associated with one set.

of DAGs) overlap, therefore, all the sets of DAGs converged to the same distribution. Hence, there is evidence in favor of the convergence of the main set of DAGs (containing two million samples). The histogram displays the scores of the DAGs as an independent variable. Completely different DAGs may have equal or slightly different scores, therefore, further analysis has to be done to understand if the 2 000 000 DAGs sampled by MDS approximate correctly the posterior distribution of DAGs.

Posterior probability of edges analysis on MDS

The posterior distribution of the probability of each edge can be easily computed when DAGs are sampled. This type of analysis gives more information on the convergence of different sets of DAGs. Let us denote by ζ a $N \times N$ matrix that contains the posterior probability of each edge (more precisely, the (i, j) entry of ζ contains the value of the probability of the presence of an edge going from X_i to X_j). Since MDS outputs a set containing sampled DAGs, $\zeta^{\text{MDS}}(t_0, T)$ depends on the starting iteration (or time index t_0) and on the final time index T that can be arbitrarily chosen. Matrix

$\zeta^{\mathrm{MDS}}(t_0, T)$ can be obtained using the equation

$$\zeta^{\mathrm{MDS}}(t_0, T) = \frac{\sum_{t>t_0}^{T} A(t)}{(T - t_0)} \tag{5.1}$$

where $A(t)$ is the adjacency matrix of the t^{th} sample $M(t)$. For practical reasons, let us use the notation without time indexes ζ^{MDS}, when t_0 refers to the first DAG (or burn-in value when MCMC algorithms are used) and T refers to the last index of the set.

To check whether the posterior distribution of the edges converged correctly to the posterior distribution, the main set has been divided into four sets of DAGs of size 500 000. The correlation of the posterior probability of the edges of the four sets is shown in Figure 5.2. The coordinates of each point represent the respective edge probability of the two sets compared. More precisely, let us assume that MDS set S and MDS set S' are compared. The correlation graph contains N^2 points (number of entries in the matrix ζ) and the coordinates of each point are $(\zeta_{i,j}^{S}, \zeta_{i,j}^{S'})$ where S and S' refer to the different MDS sets.

Figure 5.2 compares the posterior probability of the edges for all four sets. In the correlation graphs, all points are located along the diagonal (that represents the identity function). Thus, $\zeta_{i,j}^{S} \approx \zeta_{i,j}^{S'}$ for any S and S'. We can conclude that all four sets converged to the same posterior distribution, and the posterior probability of the edges ζ^{MDS}, computed using the main set containing two million DAGs sampled, can be used as a reference.

This exact sampler has already been proven to be unbiased, but it can be numerically unstable. To double-check the correctness of the posterior distribution obtained by these algorithms, the posterior probability of the edges obtained using MDS can be compared with the posterior probability of the edges obtained using BIDA.

Posterior probability of edges analysis on BIDA

BIDA algorithm does not sample DAGs but computes the posterior probability of each parents set of each node. The posterior probability of each edge, going from node X_i to X_j, can be obtained by calculating a simple ratio. In the nominator, there is the sum of the elements of the set containing the local scores of X_j of all DAGs, where X_i is the parent of X_j. In the denominator there is the sum of the elements of the set containing the local scores of X_j of all DAGs. Hence, the matrix ζ^{BIDA} that contains all the posterior probabilities of the edges can be easily computed.

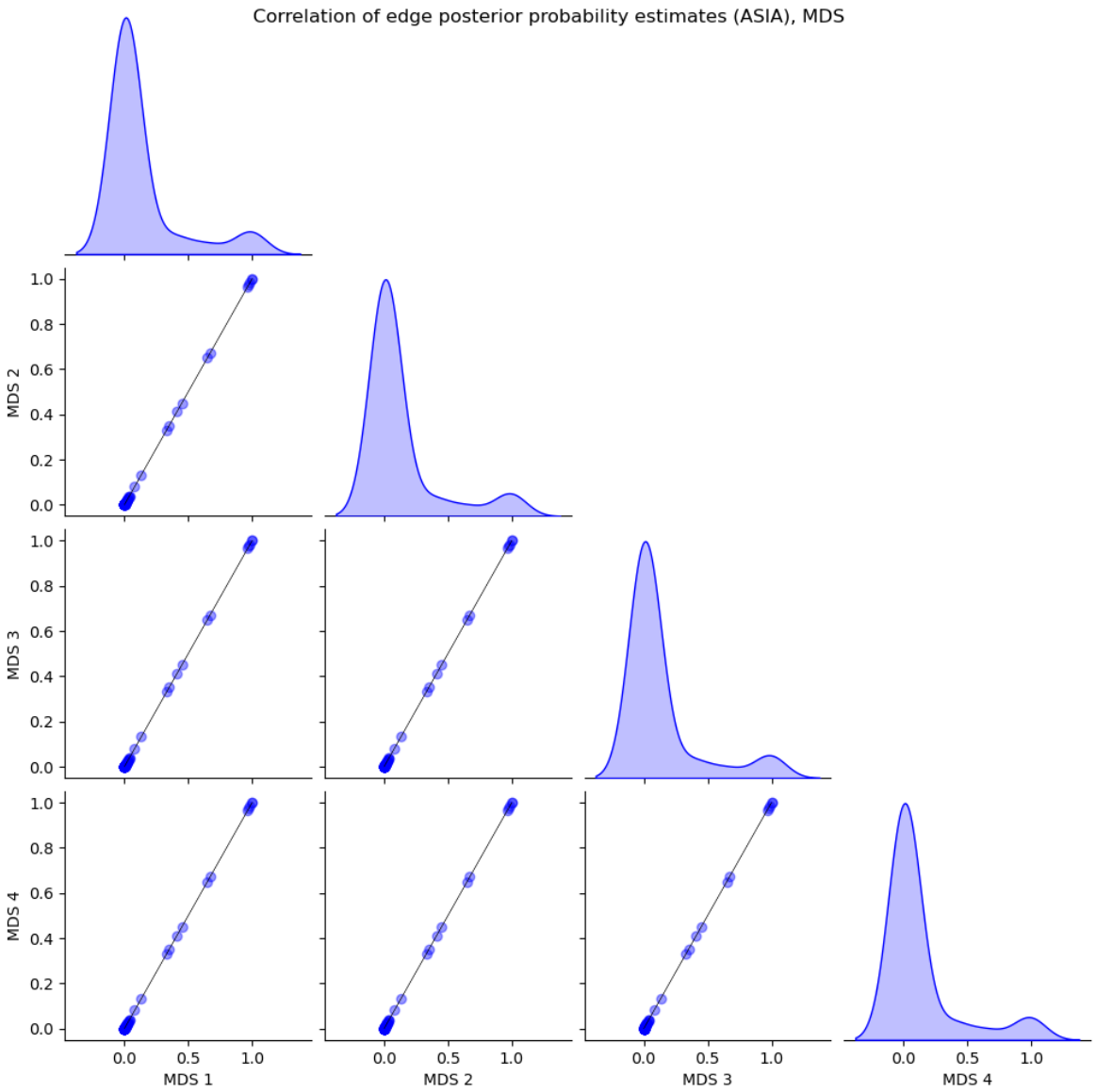

Figure 5.2: Correlation of the posterior probabilities of the edges among 4 MDS sets (exact sampling). The sets contain 500 000 samples each. Each point of any of the sub-graphs has coordinates equal to the posterior probability of the edge going from X_i to X_j for the 2 compared sets.

L1 loss for posterior probability of edges comparison

The value of ζ^{MDS} and ζ^{BIDA} can now be compared to check whether the two methods converge to the same posterior distribution. Let us define the L1 loss for the parameter ζ.

Definition 6 (L1 loss) *Given two $N \times N$ matrices ζ and ζ'. Then the* L1 *loss for the matrices ζ and ζ' is*

$$\mathrm{L1}(\zeta, \zeta') = \sum_{i=1}^{N} \sum_{j=1}^{N} |\zeta_{i,j} - \zeta'_{i,j}| \quad . \tag{5.2}$$

In our case, the entries (i, j) of ζ and ζ' represent the posterior probability of the edge going from X_i to X_j. The L1 loss has been computed using variables ζ^{MDS} (using the MDS set containing two million sampled DAGs) and ζ^{BIDA} that have been described before. For ASIA dataset, $\mathrm{L1}(\zeta^{\mathrm{MDS}}, \zeta^{\mathrm{BIDA}}) = 0.006$, therefore, we can assume that the two algorithms converged to the same posterior distribution.

For the experiments, BIDA is used as reference since the value of the posterior probability of the edges $\zeta_{i,j}$ does not depend on any time index t and it seems numerically stable enough to be used as a reference. MDS algorithm samples DAGs, therefore, there is an implicit stochasticity that depends on the number of nodes of the network, the maximum in-degree, and the number of sampled DAGs.

5.4 Limits of the structure MCMC

Some problems may arise when the structure MCMC is implemented without any parallel tempering scheme. The autocorrelation of this type of sampler is high, consequently, the DAGs sampler might not be able to cross low-probability regions, resulting in the MCMC getting stuck near local modes. As already expressed in Section 5.2, this wrong behavior of the MCMC can be seen frequently when the data points are large in number. In these cases, the posterior distribution has a high probability mass function for a small number of DAGs and a low probability mass function for the majority of DAGs. Thus, low-probability regions are extended.

5.4.1 Limits of MC^3 move

When the only moves allowed are simple MC^3 moves (adding, removing or reversing an edge) problems may arise even for small networks. Figure 5.3 shows the behavior of six different structure MCMC runs (using the ASIA dataset) without parallel tempering, without REV moves, and without MBR moves. They do not reach the same posterior distribution (some chains get stuck near a local mode). In the graph, each color is associated with an independent run. For example, the 6$^{\mathrm{th}}$ run (blue line) reached the correct highest scoring mode, but other runs got stuck near some local maxima (green, pink, and gold lines).

A practical example may be beneficial to illustrate why it is hard for the structure MCMC to escape local maxima. Let us denote by $M_{\text{blue}}(t)$ and by $M_{\text{green}}(t)$ the t^{th} DAG sampled by the 6^{th} run and by the 2^{nd} run of the structure MCMC respectively. Let us fix two time indexes

$$t_1 = 1\ 000\ 007 \quad \text{and} \quad t_2 = 1\ 000\ 025 \quad .$$

The difference in score of the DAGs belonging to the two different runs at the above-specified time indexes can be expressed as

$$\text{Score}\Big(M_{\text{blue}}(t_1)\Big) - \text{Score}\Big(M_{\text{green}}(t_2)\Big) \approx 10 \quad .$$

These specific DAGs differ of 3 edges. The DAG $M_{\text{int.}}$ is any intermediate DAG obtained by applying to $M_{\text{green}}(t_2)$ a MC^3 move to any of these 3 edges to obtain a DAG that resembles more $M_{\text{blue}}(t_1)$. The value of the difference $\text{Score}\Big(M_{\text{green}}(t_2)\Big) - \text{Score}(M_{\text{int.}}) > 10$ for every $M_{\text{int.}}$, therefore, the transition from $M_{\text{green}}(t_2)$ to $M_{\text{int.}}$ occurs with a lower probability than e^{-10}. Hence, it is rare for the chain to escape from the local mode. However, the process is ergodic, therefore, in a theoretical scenario where infinite DAGs are sampled, a structure MCMC that uses only MC^3 moves generates samples according to the posterior distribution. In practice, the number of sampled DAGs is limited, and the posterior distribution generated by this type of structure MCMC is frequently biased.

Adding REV and MBR moves to increase the performance

REV moves and MBR moves can be useful, in some cases, to cross low-probability regions. By adding these moves, the number of neighbors of any single DAG increases, creating new chances of escaping local maxima. The increase of performance of the structure MCMC on the ASIA dataset, when the 3 moves are available, can be seen in Figure 5.4. The probability of performing a REV move has been set to 0.7, while for the MBR move, the probability has been set to 0.2, after hyperparameter tuning (explained in detail in Section 5.4.2), to describe the best-case scenario for a structure MCMC with this set of moves, without parallel tempering. All the 6 structure MCMC reach the main mode of scores in less than 1000 iterations.

5.4.2 Limits of REV and MBR moves

A simple structure MCMC, that samples using MC^3, REV, and MBR, might not be a suitable algorithm when the number of nodes N increases. An MCMC run might get

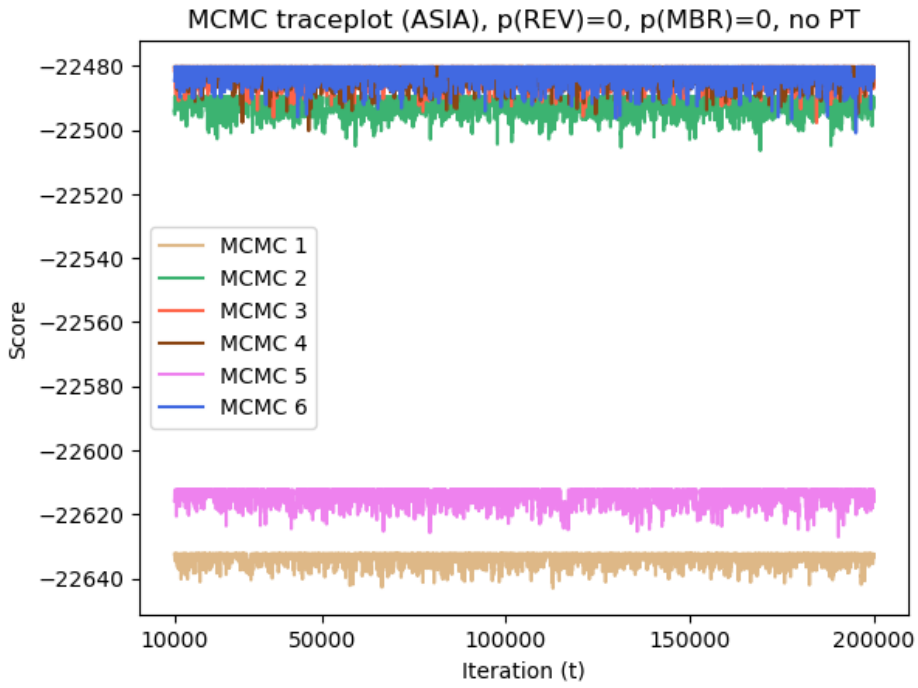

Figure 5.3: Traceplots of 6 structure MCMC runs (ASIA) that use only MC^3 moves. Each chain contains 200 000 samples, and the first 10 000 samples have been discarded (burn-in).

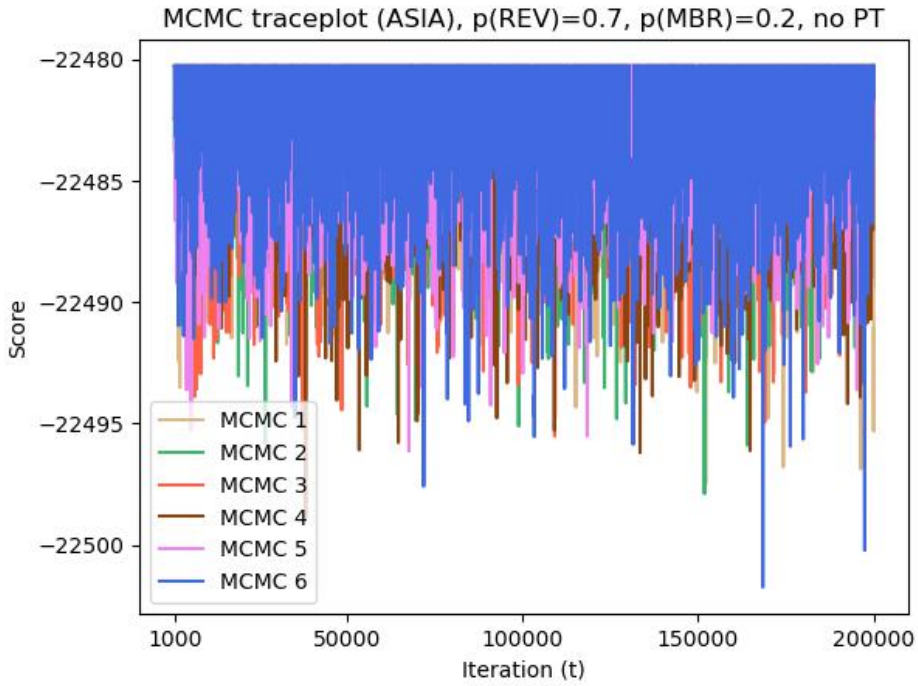

Figure 5.4: Traceplots of 6 structure MCMC runs (ASIA) using $p(MC^3) = 0.1$, $p(\text{REV}) = 0.7$ and $p(\text{MBR}) = 0.2$. Each chain contains 200 000 samples, and the first 1 000 samples have been discarded (burn-in).

stuck around a set of DAGs, even if the proposed DAG differs of more than one edge

compared to the starting DAG. This behavior can be easily seen in Figure 5.5. Three structure MCMC without parallel tempering, containing 1 000 000 DAGs, have been implemented to analyze the INS dataset. The same optimal probabilities for each move have been used. Two of the three different MCMC runs reach the main score mode,

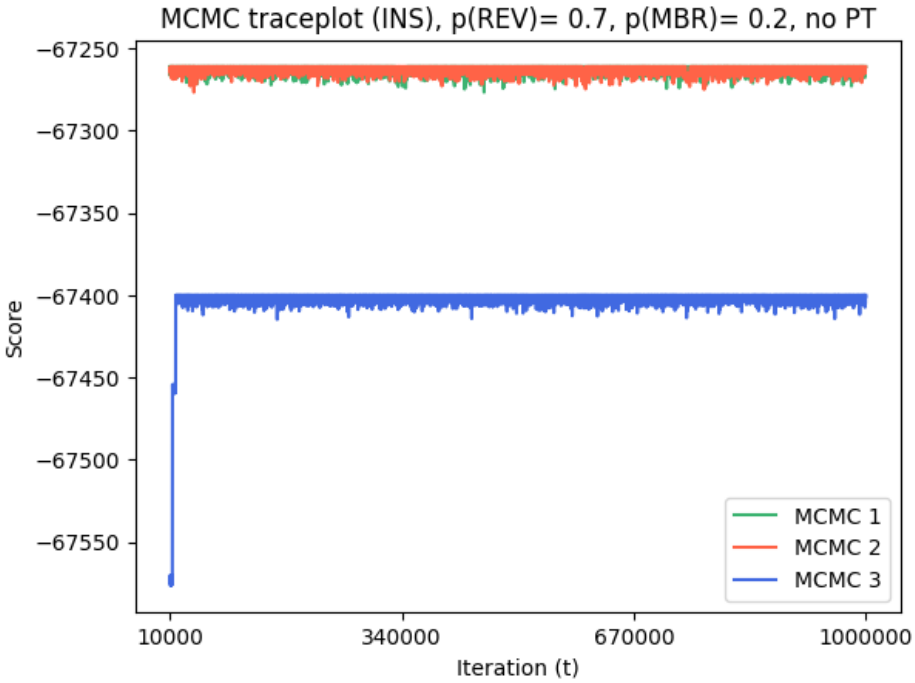

Figure 5.5: Traceplots of 3 structure MCMC runs (INS) using $p(\text{MC}^3) = 0.1$, $p(\text{REV}) = 0.7$ and $p(\text{MBR}) = 0.2$. Each chain contains 1 000 000 samples and the first 10 000 samples have been discarded (burn-in).

while one gets stuck near a local mode that is significantly lower in score compared to the main one. The difference in score among the two modes is around 150, therefore the DAGs that are in MCMC 1 and MCMC 2 have a probability mass function that is around e^{150} larger than the probability mass function of a DAG that is in MCMC 3. Hence, the structure MCMC without parallel tempering cannot be reliable for sampling in more challenging situations.

5.4.3 Limits of score and traceplot analysis

Traceplot analysis is not a sufficient measure of convergence and problems regarding the lack of information caused by this type of analysis can be seen by looking at the behavior of the structure MCMC applied to the MUSH dataset. Three structure MCMC runs, without parallel tempering and with the same hyperparameters for the moves of the previous experiments, have been implemented.

Figure 5.6 shows that the different samplers struggle to reach the main score mode, but eventually they manage to sample DAGs with similar scores after around 400 000 iterations. Hence, at first glance, by looking only at the traceplots of the different runs, after setting a suitable burn-in value of 500 000 DAGs, the hypothesis of convergence of the three different runs to the correct posterior distribution seems reasonable.

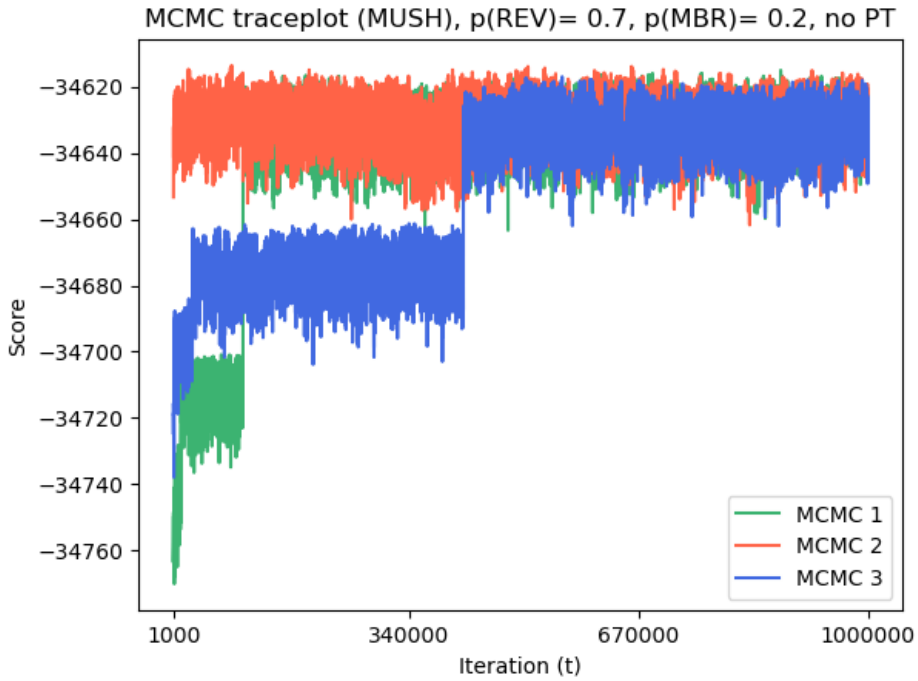

Figure 5.6: Traceplots of 3 structure MCMC runs (MUSH) using $p(\text{MC}^3) = 0.1$, $p(\text{REV}) = 0.7$ and $p(\text{MBR}) = 0.2$. Each chain contains 1 000 000 samples and the first 10 000 samples are being discarded (burn-in).

In reality, convergence is not achieved. This can be checked by computing the posterior probability of the edges for all the runs and comparing them to the true values of the posterior probability of the edges obtained using BIDA algorithm. By looking at Figure 5.7, we can see that most of the posterior probabilities of the edges, obtained by the DAGs sampled in different structure MCMC runs, differ from those generated by the reference algorithm. Therefore, the posterior distribution generated by the structure MCMC without parallel tempering is biased. The sets of DAGs sampled by the structure MCMC show more correlation among themselves, giving a clue that they might get stuck near similar local modes and they are not able to explore well the posterior distribution.

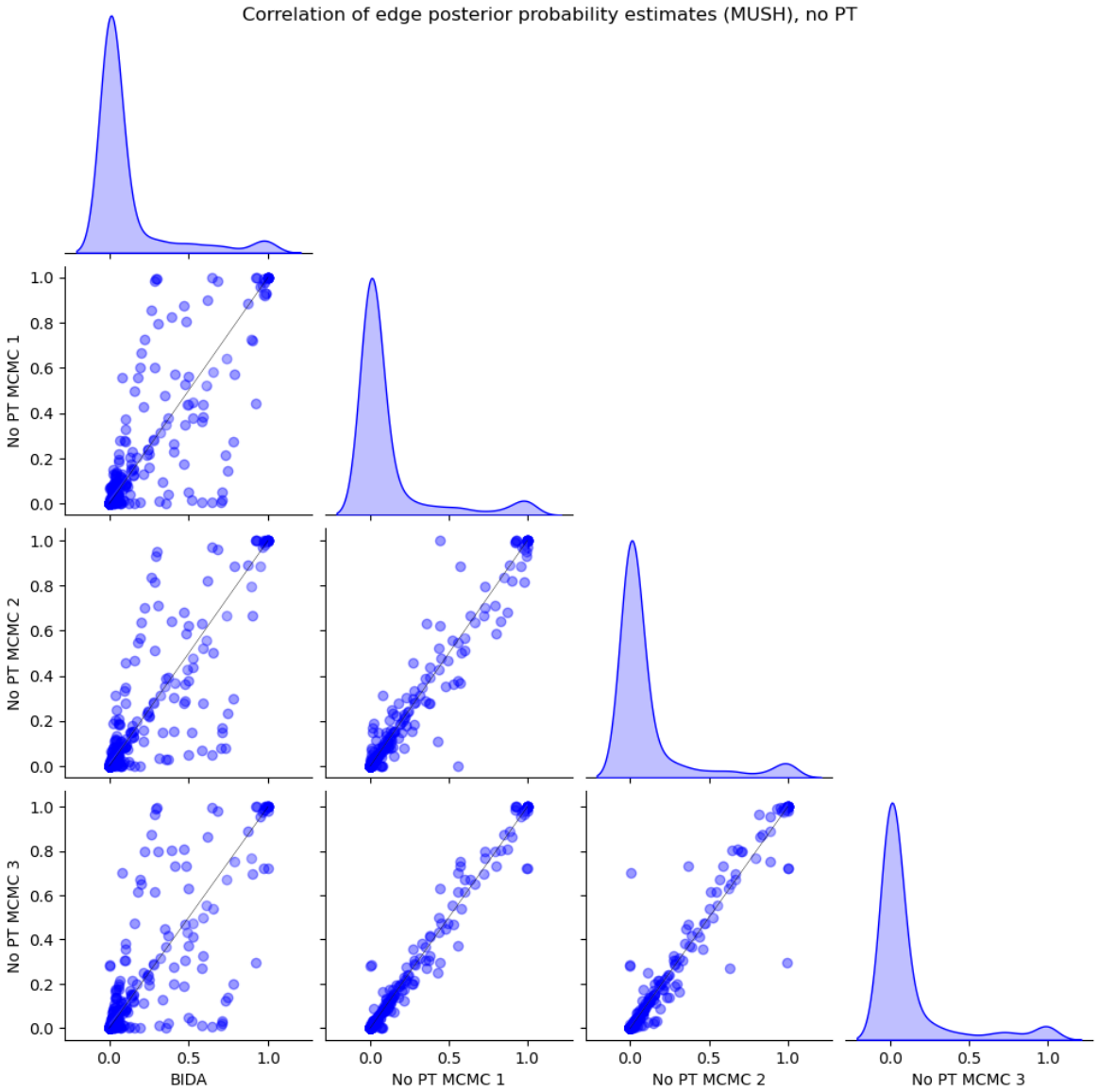

Figure 5.7: Correlation of the posterior probabilities of the edges (MUSH) among BIDA (first column) and 3 structure MCMC runs with $p(\text{REV}) = 0.7$, $p(\text{MBR}) = 0.2$ and no PT scheme implemented. The structure MCMC algorithms sampled 1 000 000 DAGs. The first 500 000 DAGs have been discarded. Each point of any sub-graph has coordinates equal to the posterior probability of the presence of an edge going from X_i to X_j for the two compared algorithms.

5.5 Properties of the structure MCMC moves and hyperparameter tuning

Recalling the theory described in Section 3.4, an exploration kernel that performs the three different moves (MC^3, REV, and MBR) with different probabilities approaches the stationary distribution without generating any bias. In the previous section, it was shown that a structure MCMC, which performs only MC^3, has strong autocorrelation,

therefore, the sampler might not be able to cross low-probability regions. REV moves and MBR moves generally perform better than MC^3 moves because they increase the number of neighbors of any DAG, keeping the acceptance ratio of the proposed DAG reasonably high.

5.5.1 Comparing REV and MBR moves

To check the performance of those two moves, several structure MCMC were implemented without any parallel tempering scheme, using the ASIA dataset. Three of them can perform only REV moves while other three can perform only MBR moves. The traceplot of the different runs is shown in Figure 5.8. It is clear that samplers that use only REV moves perform better than samplers that use only MBR moves, since the first ones reach the main score mode fast while the latter ones struggle more to propose high-scoring DAGs and may get stuck near a local mode. Intuitively, this result is expected, since the REV move involves adding, removing, and reversing edges in a single move, while MBR cannot reverse any edge in a single move.

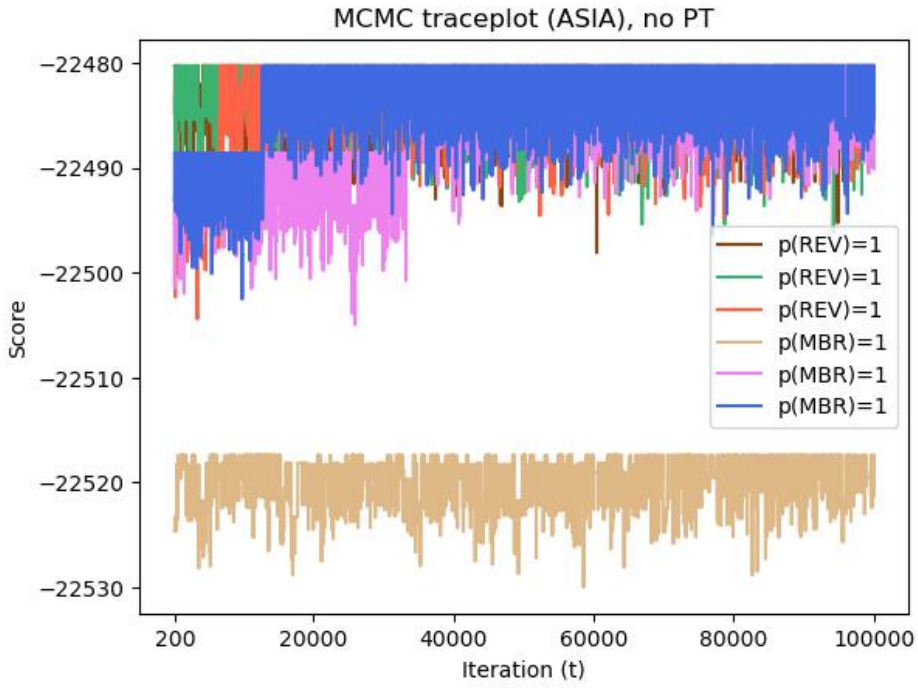

Figure 5.8: Traceplot of 6 structure MCMC runs (ASIA). 3 MCMC algorithms can perform only REV moves, while the other 3 can perform only MBR moves. Each chain contains 100 000 samples, and the first 200 samples have been discarded to show the behavior of the samplers, when they approach the main score mode.

5.5.2 Finding optimal probabilities for each move

Su and Borsuk [8] suggest that a good value for the probability of performing REV and MBR moves is 1/15 for both. However, no experiments are shown to support this statement. In addition, the reference distribution that has been used in the experiments has been generated using a structure MCMC initialized at a high-scoring DAG. This method may generate bias since a simple structure MCMC, which performs MC^3 moves with high probability, frequently, does not approximate well the posterior distribution.

Figure 5.7 shows that none of the structure MCMC runs converges to the posterior distribution, therefore, they cannot be used as reference, even though they sample high-scoring DAGs. If the reference distribution is biased, and assuming that the REV and MBR moves would help to escape a local mode, hyperparameter tuning fails since it would prefer biased MCMC runs, reducing the value of the probability of the moves that would reduce the bias (in this case the REV move and the MBR move with less impact). Hyperparameter tuning has to be performed using a correct reference distribution and in challenging situations (for example, when the likelihood term has strong influence on the posterior distribution) to ensure good flexibility for the model.

In the experiments, hyperparameter tuning for the probability of the moves has been performed using the ASIA dataset since it is associated to a challenging posterior distribution to sample from and, at the same time, the number of nodes is small. These features of the dataset lead to consistent results with a reasonable computational cost (since grid search has to be performed). The L1 loss has been used as loss function to minimise and ζ^{BIDA} has been used as reference. The loss function has been averaged over 100 DEO structure MCMC runs where the same parameters are used (only the initial seed is changed). Each MCMC run contains 10 000 samples and the first 5 000 have been discarded (burn-in).

The best parameters for the probability of the moves for ASIA, obtained after hyperparameter tuning, are $p(\text{REV}) = 0.7$ and $p(\text{MBR}) = 0.2$. Therefore, simple moves are performed with probability equal to 0.1. There is no theoretical proof that shows that these hyperparameters guarantee an efficient sampling also for other networks, but, in practice, they generalize well and outperform significantly the original probabilities suggested by Su and Borsuk.

5.6 Comparing PT algorithms

As already mentioned in the previous section, most of the experiments have been performed using the ASIA dataset, since it can provide reliable comparisons, while keeping the computational cost low. The optimal values for the probabilities of performing each move have been set according to the results of hyperparameter tuning.

5.6.1 Description of the algorithms

The parallel tempering algorithms that have been used in the experiments are the non-dynamic DEO structure MCMC, the non-dynamic SEO structure MCMC, the non-dynamic SRS structure MCMC, the non-tuned DEO structure MCMC and the dynamic DEO structure MCMC. The *non-dynamic* attribute refers to a structure MCMC where the two tuning phases are performed, but the second tuning phase is stopped after a fixed number of 8 tuning steps. The number of DAGs sampled in each of the second tuning steps is set to 3 000 while the total number of iterations for the first tuning phase is set to 2 000. In the *dynamic* sampler, the second tuning phase is not stopped.

The starting number of chains has been set to 60 and, after the first tuning phase, it has been reduced to an optimal number $C + 1 = 13$ by the tuning algorithm for all samplers. Therefore, for the *non-tuned* DEO sampler, the number of chains $C + 1$ has been set to 13 to guarantee a fair comparison. Its annealing schedule is built using a popular parallel tempering scheme, where the temperatures are obtained using the equation $\beta_{C-c} = 1/2^i$ for $c \in \{0, \dots, C-1\}$ and $\beta_0 = 0$.

The performance of the above-mentioned samplers is compared also to the performance of the structure MCMC that does not have any parallel tempering scheme implemented. The traceplots of a single run of each algorithm are shown in Figure 5.9. All algorithms reach fast the main score mode and display a similar behavior after discarding the burn-in samples.

5.6.2 Normalized-L1 loss and max-loss

To compare the performance of each algorithm, let us introduce two different measures: the normalized-L1 loss and the max-loss.

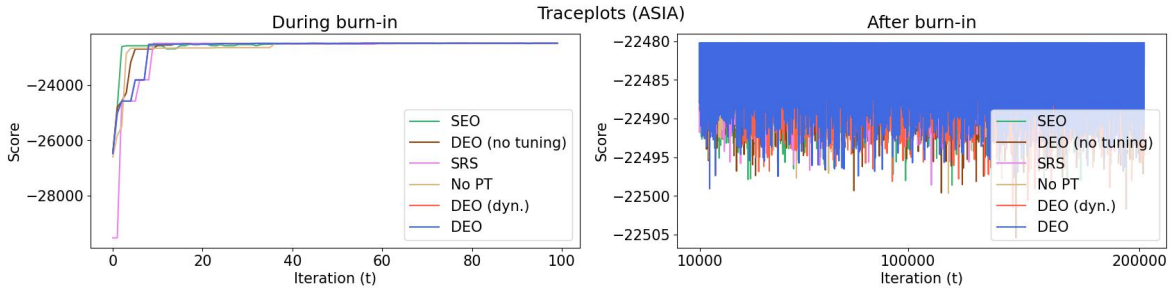

Figure 5.9: Traceplots of different structure MCMC algorithms (ASIA). $p(\text{REV}) = 0.7$ while $p(\text{MBR}) = 0.2$ for all samplers. Each MCMC contains 200 000 samples and the first 10 000 samples have been discarded (burn-in value).

Defining the losses

The L1 loss cannot be used as an absolute indicator of convergence, since it depends on the expected value of edges of the posterior distribution. A structure MCMC that is biased from the reference, and that samples from a posterior distribution where the expected number of edges is small, may result in a smaller L1 loss, compared to the L1 loss of a less biased structure MCMC, that samples from a posterior distribution where the expected number of edges has a larger value. To overcome this problem, the normalized-L1 loss has been defined.

Definition 7 (Normalized-L1 loss) *Given two $N \times N$ matrices ζ and ζ', where ζ' serves as the reference for the parameter. Then the* normalized-L1 *loss is defined as*

$$\text{normalized-L1}(\zeta, \zeta') = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}|\zeta_{i,j} - \zeta'_{i,j}|}{\sum_{i=1}^{N}\sum_{j=1}^{N}\zeta'_{i,j}} \quad . \tag{5.3}$$

When ζ represents the matrix that contains the posterior probabilities of the edges, the denominator of Equation 5.3 represents the expected number of edges in the network. The normalized-L1 loss can be computed for ASIA since we have a reference value ζ' for the parameter ζ, which has been computed using BIDA algorithm.

The max-loss for two models, on the other hand, refers to the biggest value of the set containing the absolute difference of the posterior probability of all the edges of the two compared models. More precisely

Definition 8 (Max-loss) *Given two $N \times N$ matrices ζ and ζ'. Then the* max-loss *for the matrices ζ and ζ' is*

$$\text{max-loss}(\zeta, \zeta') = \text{max entry of matrix } |\zeta - \zeta'| \quad . \tag{5.4}$$

The max-loss gives a practical measure of the quality of the model. If the value of the max-loss is bigger than 0.5, it means that, on average, a random number generator between 0 and 1 would give a more accurate value of the posterior probability of that edge. Also in this case, the reference ζ' is obtained using BIDA algorithm. For practical reasons, in the formulas above, $\zeta = \zeta(t_0, T)$ is a value (since t_0 refers to the burn-in value and T refers to the last time index of the MCMC), therefore, losses are also values. Defining $\zeta(t_0, t)$ as a function of t, the above-mentioned losses become functions of t too.

Empirical results

The normalized-L1 loss functions and the max-loss functions, of the different DAGs samplers are shown in Figure 5.10. Three structure MCMC are run for each algorithm and the averaged value of the loss of the three runs is reported as a function of the time index t (which refers to the iteration). Log-scale for both x-axis and y-axis is used to show that the losses are still decaying after 200 000 iterations.

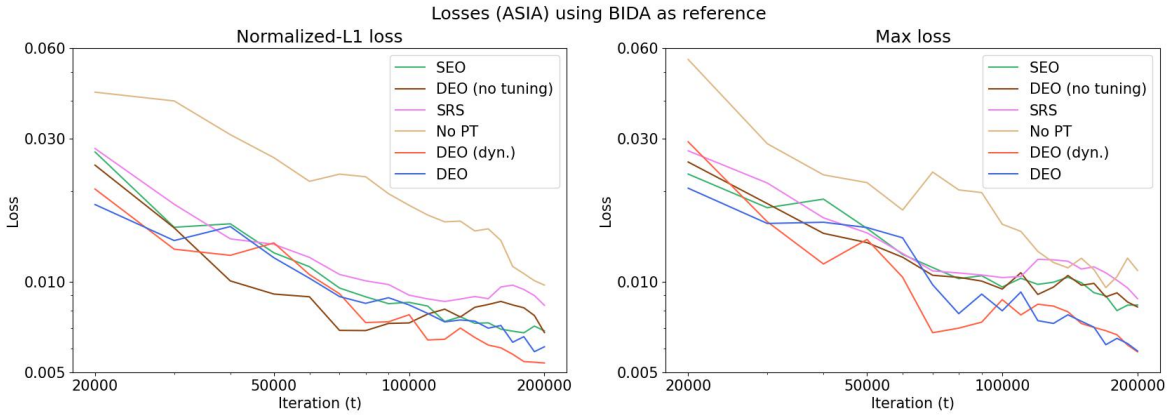

Figure 5.10: Normalized-L1 loss and max-loss of the different structure MCMC algorithms (ASIA). p(REV)=0.7 while p(MBR)=0.2 for all the samplers. Averaged value over 3 structure MCMC runs containing 200 000 samples. The first 20 000 DAGs have been discarded (burn-in).

The structure MCMC without parallel tempering performs poorly compared to the ones that have a parallel tempering scheme implemented. This was expected since the structure MCMC may get stuck near local maxima, if no PT scheme is

Table 5.1: Losses (ASIA) after 200 000 iterations using BIDA as reference.

Algorithm	Normalized-L1 loss	Max-loss
DEO	0.0061	0.0059
SEO	0.0069	0.0084
DEO (no tun.)	0.0068	0.0087
SRS	0.0084	0.0083
No PT	0.0098	0.0082
DEO (dyn.)	**0.0054**	**0.0058**

implemented, while, in theory, PT allows a correct exploration of the low-probability regions. However, parallel tempering involves a higher computational cost, therefore, the comparison is not completely fair.

DEO algorithm performs better compared to the other parallel tempering algorithms, even though the difference in loss is not extremely significant. It is clear that the tuning phases are beneficial, since the DEO structure MCMC with fixed annealing schedule performs poorly compared to the ones where tuning is performed.

The dynamic tuning scheme performs slightly better compared to the standard one, but it might be the result of the random exploration of the posterior distribution, therefore, no clear statement can be made, regarding the utility of dynamic tuning. In Table 5.1 the averaged value, over three different runs (after 200 000 iterations), of the normalized-L1 loss and of the max-loss are reported, for each of the above-mentioned algorithms. In Section 5.7, experiments done in more challenging settings show the improvement of performance connected to the implementation of dynamic tuning.

5.6.3 Round trips as measure of performance

To measure the efficiency of communication among the chains of the different parallel tempering algorithms, the number of round trips performed has been measured. A higher number of round trips occurs when the parallel tempering scheme allows better communication among the contiguous chains.

In Table 5.2, the averaged value of the number of round trips is shown for the different parallel tempering schemes, using the ASIA dataset. DEO algorithm outperforms the other schemes, also in terms of round trips performed. The *non-tuned* geometric scheme does not perform any round trip, since some chains are not able to communicate well. More specifically, the chains at temperatures β_0 and β_1 do not communicate at all. In practice this limitation is not affecting extremely negatively the

Table 5.2: Number of round trips (ASIA) averaged over 3 different runs

Algorithm	$\mathbb{E}$(round trips)
DEO	1100
SEO	917
DEO (no tun.)	0
SRS	360.33
DEO (dyn.)	**1117**

performance of the algorithm since the posterior distribution of the edges, obtained by sampling using this fixed temperature scheme, is fairly good.

The number of round trips performed using the dynamic tuning scheme is higher compared to the one obtained by using a DEO scheme without dynamic tuning. On the other hand, the difference is small, therefore, also in this case, it can be a consequence of the random exploration of the MCMC runs.

5.7 Dynamic tuning performance

In the previous section, the experiments on the ASIA dataset showed that the DEO scheme outperforms the other parallel tempering schemes. The normalized-L1 loss and the max-loss drop faster when the DEO scheme is used and the number of round trips performed is greater compared to the SEO and SRS schemes (as expected from the theory in Section 4.4).

No clear statement could be made regarding the utility of the dynamic tuning phase in ASIA experiments, since the performance of the DEO algorithm was not significantly dependent on the dynamism of the algorithm. In this section, further experiments are described, using more challenging networks, to show how dynamic tuning improves the performance of the structure MCMC.

5.7.1 Parameters of the structure MCMC for the different datasets

Each experiment requires different parameters of the structure MCMC since the number of nodes and the maximum in-degree varies for each dataset. More precisely, the number of iterations for the first tuning phase and the starting number of chains $C+1$ depend on the complexity of the network. There are some standard parameters that have been fixed for each experiment. The optimal probability of the moves are set

to the values obtained after hyperparameter tuning, described in Section 5.5.2. The number of iterations of each tuning step of the second tuning phase is set to 3 000. The second tuning phase is stopped for the not dynamic algorithms, after 8 tuning steps. For each network analyzed, a different value for T is used. The number of sampled DAGs needed for a correct convergence of the MCMC depends on the number of nodes, on the maximum in-degree, and on the shape of the posterior distribution.

MUSH parameters

As already expressed in Section 5.2, MUSH is a complex dataset to analyze. The number of nodes is $N = 23$ and the maximum in-degree is four, therefore, the space of DAGs is large. The first tuning phase, of each structure MCMC, consists of 20 000 iterations. Subsequently, 1 500 000 DAGs are sampled. The starting number of chains is $C + 1 = 100$, while the optimal number of chains found by the tuning algorithm, for all runs, after the first tuning phase, is $C + 1 = 36$.

Dynamic tuning has been enabled for all the three different parallel tempering schemes and the performance of the dynamic samplers is compared to the performance of the DEO sampler without dynamic tuning (where the annealing schedule is updated, in the second tuning phase, 8 times, after 3 000 iterations). Additional structure MCMC runs, without any parallel tempering scheme, have been implemented as a further comparison. To have a more reliable result, four structure MCMC have been run for each algorithm. The traceplots of the different algorithms show that MCMC runs converged to the highest scoring mode correctly.

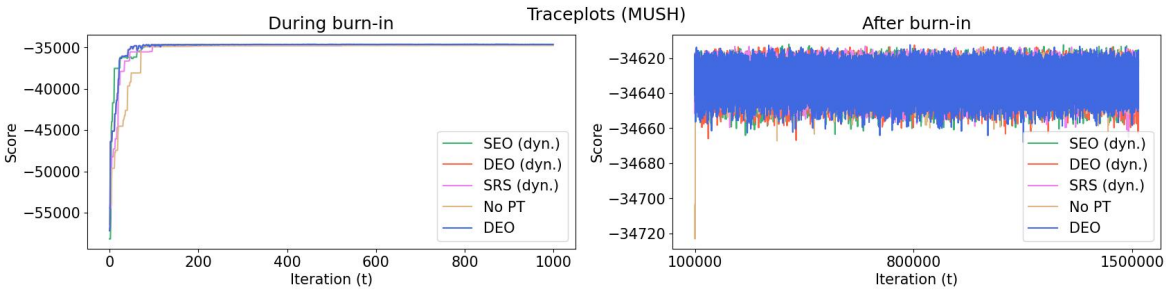

Figure 5.11: Traceplots of different structure MCMC algorithms (MUSH). Each MCMC contains 1 520 000 samples and the first 100 000 samples have been discarded (burn-in value).

ZOO parameters

The ZOO dataset contains only 101 data points, therefore, the posterior distribution does not present extremely low-probability regions between modes. 1 000 000 DAGs

have been sampled, after a first tuning phase of 20 000 iterations. The starting number of chains has been set to $C + 1 = 100$ and, after the first tuning phase, it has been reduced to $C + 1 = 16$.

In the experiments involving ZOO dataset, only the dynamic DEO structure MCMC and the non-dynamic DEO structure MCMC are compared, and five different runs are implemented for each algorithm. Figure 5.12 shows that the traceplots of the different MCMC runs converge around the same score mode.

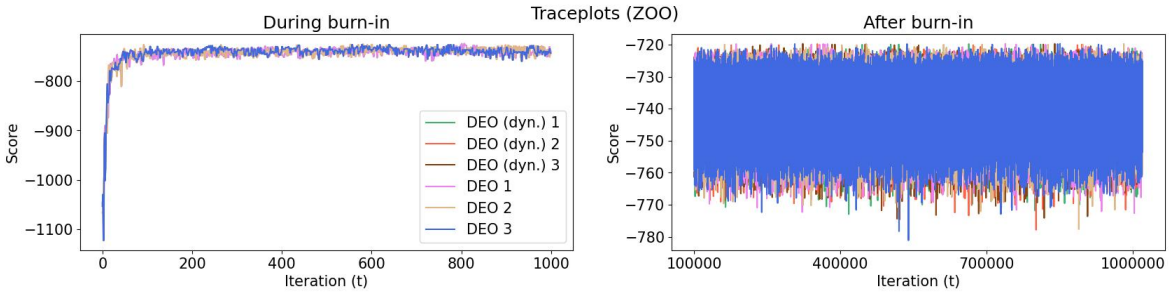

Figure 5.12: Traceplots of dynamic and non-dynamic DEO structure MCMC runs (ZOO). Each MCMC contains 1 020 000 samples and the first 100 000 samples have been discarded (burn-in value).

INS parameters

In the experiments involving the INS dataset, similarly to the ZOO experiments, only the performance of the dynamic and non-dynamic DEO are compared. Five structure MCMC runs, containing 530 000 DAGs, have been implemented for both algorithms. The first tuning phase consists of 30 000 iterations in total. The starting number of chains is $C + 1 = 100$ and after tuning, it has been reduced to 36 (the same number of chains that was obtained for MUSH experiments). Figure 5.13 shows that both algorithms correctly converge to the highest scoring mode.

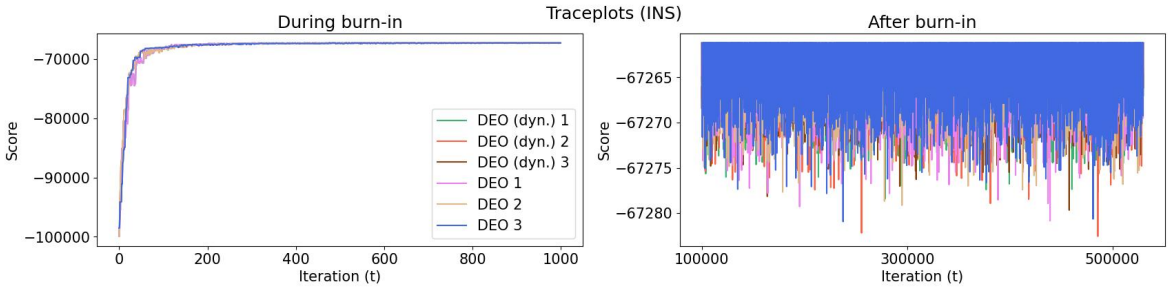

Figure 5.13: Traceplots of dynamic and non-dynamic DEO structure MCMC runs (INS). Each MCMC contains 530 000 samples and the first 100 000 samples have been discarded (burn-in value).

5.7.2 Normalized-L1 loss and max-loss comparisons

BIDA reference distribution is available for networks that have $N < 24$ number of variables, hence, the normalized-L1 loss and max-loss functions can be computed and used as a measure of performance not only for ASIA, but also for MUSH and ZOO.

MUSH losses

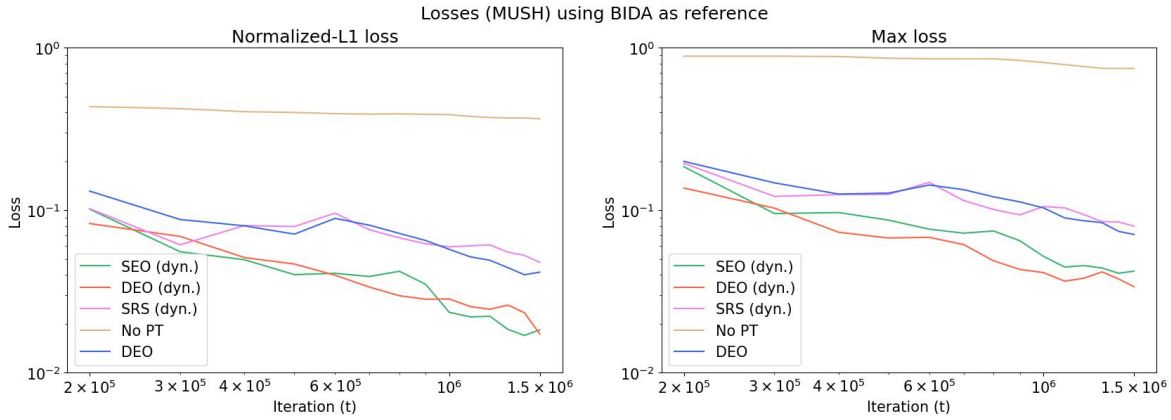

Figure 5.14: Normalized-L1 loss and max-loss of the different structure MCMC algorithms (MUSH). $p(\text{REV}) = 0.7$ while $p(\text{MBR}) = 0.2$ for all samplers. Averaged value over 4 structure MCMC runs containing 1 520 000 samples. The first 100 000 DAGs have been discarded (burn-in)

Figure 5.14 shows that the DEO and the SEO structure MCMC, with dynamic tuning scheme enabled, outperform the DEO structure MCMC with a non-dynamic annealing schedule (fixed after tuning). The performance of the non-dynamic DEO sampler is comparable to the one of the dynamic SRS sampler. Formulas 4.19, 4.20, and 4.21 show that the SRS scheme, theoretically, performs poorly compared to the DEO scheme. Hence, there is strong evidence to support the increase in performance that the dynamic tuning scheme involves. Using the logarithmic scale for both axes, it is clear that the losses are still declining and do not converge to a fixed value larger than 0.

Table 5.3 shows the losses after sampling 1 500 000 DAGs using ζ^{BIDA} as reference. The averaged max-loss for the structure MCMC without parallel tempering is 0.75 after 1 500 000 iterations. It is one order of magnitude higher than the one obtained by the less efficient sampler with a parallel tempering scheme implemented. The difference of performance can be easily seen by comparing the correlation, over different runs, of the posterior probability of the edges of different dynamic DEO structure MCMC runs (Figure 5.15) and the correlation, over different runs, of the posterior probability of the edges of different structure MCMC runs without any parallel tempering scheme implemented (Figure 5.7). In Figure 5.15, all the points

stand on the diagonal of the sub-plots, therefore, the values of the posterior probability of the edges are almost equal for all the runs and they coincide with the values of the reference distribution, obtained using BIDA algorithm.

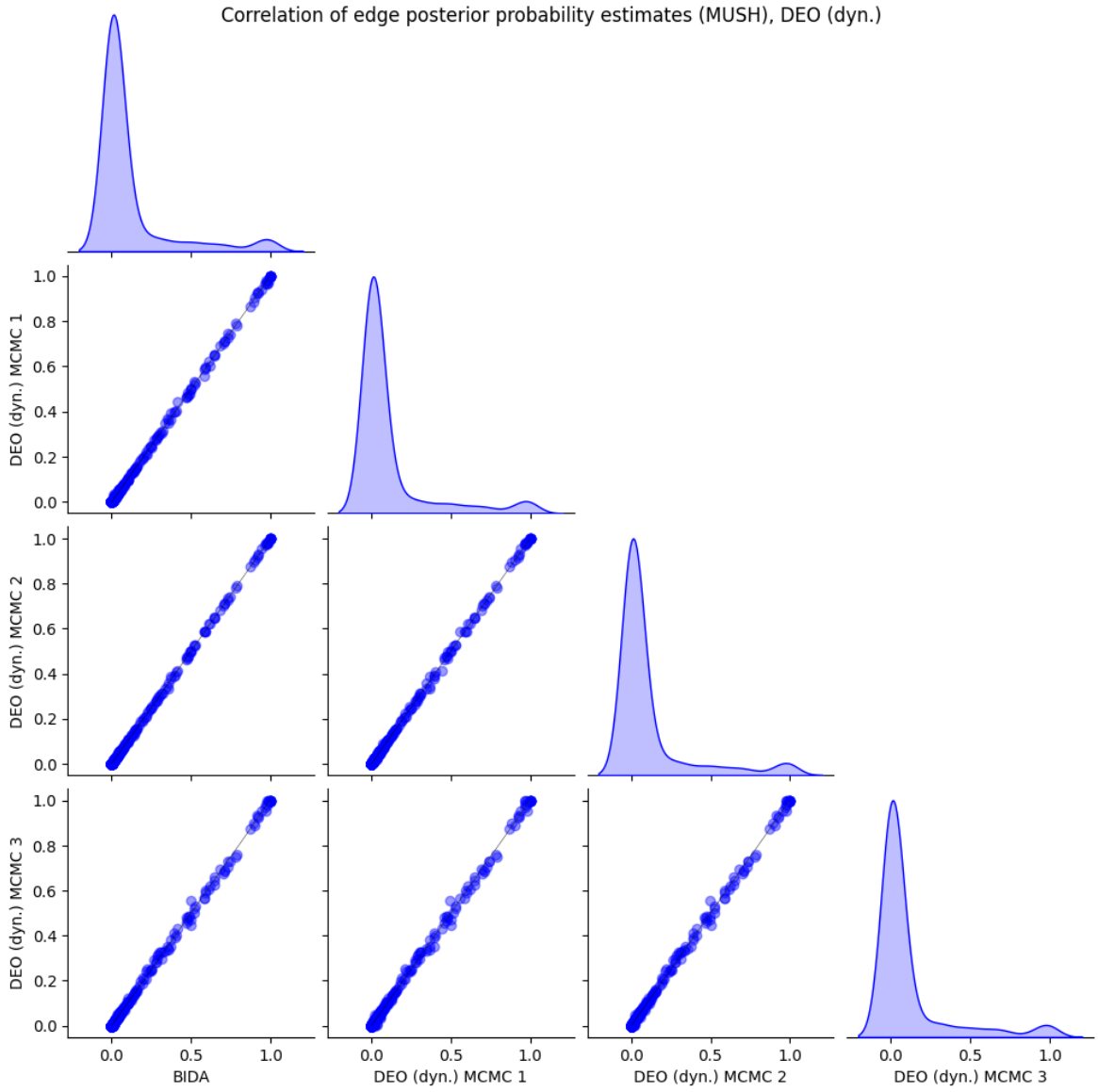

Figure 5.15: Correlation of the posterior probabilities of the edges (MUSH) among BIDA (first column) and 3 dynamic DEO structure MCMC runs with $p(\text{REV}) = 0.7$ and $p(\text{MBR}) = 0.2$. The structure MCMC algorithms sampled 1 520 000 DAGs. The first 100 000 DAGs have been discarded. Each point of any sub-graph has coordinates equal to the posterior probability of the presence of an edge going from X_i to X_j for the 2 compared algorithms.

It is true that each iteration of the parallel tempering algorithm is computationally more demanding, compared to a single iteration of the simple structure MCMC without parallel tempering. However, by looking at Figure 5.14, which shows the be-

Table 5.3: Losses (MUSH) after 1 500 000 iterations using BIDA as reference. The burn-in value is 100 000.

Algorithm	Normalized-L1 loss	Max-loss
DEO	0.042	0.071
DEO (dyn.)	**0.017**	**0.034**
SEO (dyn.)	0.018	0.042
SRS (dyn.)	0.048	0.08
No PT	0.37	0.75

havior of the losses as functions of the iteration t, we can see that the losses of the DEO dynamic structure MCMC, which sampled 100 000 DAGs, are around 6 times smaller than the losses of a simple structure MCMC without PT, that sampled 1 400 000 DAGs. These results underline the increase of performance obtained by the dynamic DEO structure MCMC and the importance of parallel tempering, in general, in the BN setting.

ZOO losses

Figure 5.16 shows that the dynamic DEO structure MCMC performs slightly better compared to the non-dynamic one. Sampling DAGs using the ZOO dataset, according to the posterior distribution, is not a challenging task, since the likelihood term is not strong and the prior term still has a strong influence on the posterior distribution. Hence, the difference in the performance of the two algorithms is small. Table 5.4 shows the value of the losses after 1 000 000 iterations, using BIDA as a reference.

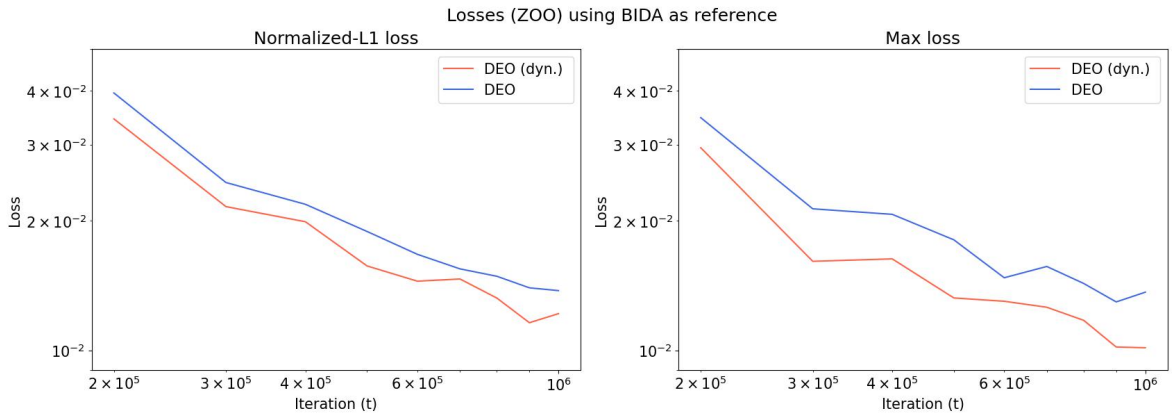

Figure 5.16: Normalized-L1 loss and max-loss for the dynamic and non-dynamic DEO structure MCMC (ZOO). $p(\text{REV}) = 0.7$ while $p(\text{MBR}) = 0.2$ for all samplers. Averaged value over 5 structure MCMC runs containing 1 020 000 samples. The first 100 000 DAGs have been discarded (burn-in).

Table 5.4: Losses (ZOO) after 1 000 000 iterations using BIDA as reference. The burn-in value is 100 000.

Algorithm	Normalized-L1 loss	Max-loss
DEO	0.014	0.014
DEO (dyn.)	**0.012**	**0.010**

Further analysis on ASIA losses

In the experiments reported in Section 5.6, 200 000 DAGs have been sampled for each MCMC algorithm, using the ASIA dataset. No clear difference in performance was found between the dynamic and non-dynamic DEO structure MCMC. Another experiment was performed to compare the efficiency of the above-mentioned samplers, using longer MCMC runs. Four structure MCMC runs, containing 2 000 000 DAGs, were implemented for the dynamic and non-dynamic algorithms, keeping fixed the other hyperparameters of the previous experiment.

Figure 5.17 shows the losses of the algorithms. Even in this case, with a higher

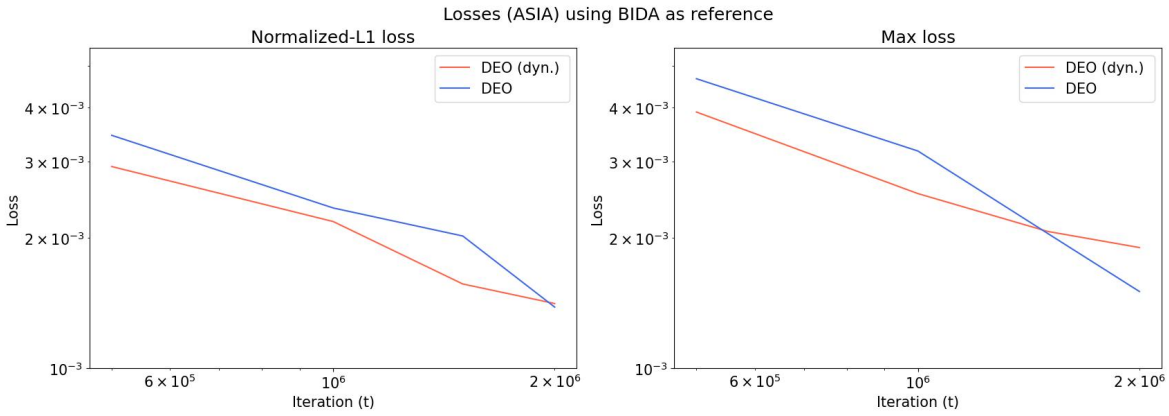

Figure 5.17: Normalized-L1 loss and max-loss for the dynamic and non-dynamic DEO structure MCMC (ASIA). $p(\text{REV}) = 0.7$ while $p(\text{MBR}) = 0.2$ for all samplers. Averaged value over 4 structure MCMC runs containing 2 000 000 samples.

number of sampled DAGs for each structure MCMC, there is not a clear difference in the performance of the dynamic DEO scheme and the non-dynamic DEO scheme. The latter one seems to perform slightly better, but also in this case, the variation of the value of the losses might be a consequence of the random exploration of the MCMC runs. Indeed, the dynamic sampler was performing better up to 1 500 000 iterations. In Table 5.5, the averaged losses of the algorithms, after sampling 2 000 000 DAGs, are reported.

Table 5.5: Losses (ASIA) after 2 000 000 iterations using BIDA as reference.

Algorithm	Normalized-L1 loss	Max-loss
DEO	0.0014	**0.0015**
DEO (dyn.)	0.0014	0.0019

INS losses

For INS dataset, a reference distribution is not available because the number of nodes exceeds the limits of BIDA and MDS. Therefore, only comparisons among the posterior probability of the edges, obtained by different structure MCMC runs, can be made. To inspect the performance of the algorithms, a new approach has to be chosen.

For each MCMC run of an algorithm (dynamic or non-dynamic), instead of using only one reference (ζ^{BIDA}), let us build a relative loss function for each of the other MCMC runs, using as references ζ^{MCMC_k} (where k is the index of the MCMC run). In the experiments described above, which involved other datasets, the total number of comparisons for each algorithm is equal to the number of MCMC runs, as each run uses only BIDA as a reference. For INS experiments, the total number of relative loss comparisons for each algorithm is equal to the factorial of the number of MCMC implemented and an averaged value of the relative loss functions is computed. In this particular case, the L1 loss is computed instead of the normalized-L1 loss, as the expected number of edges is not known.

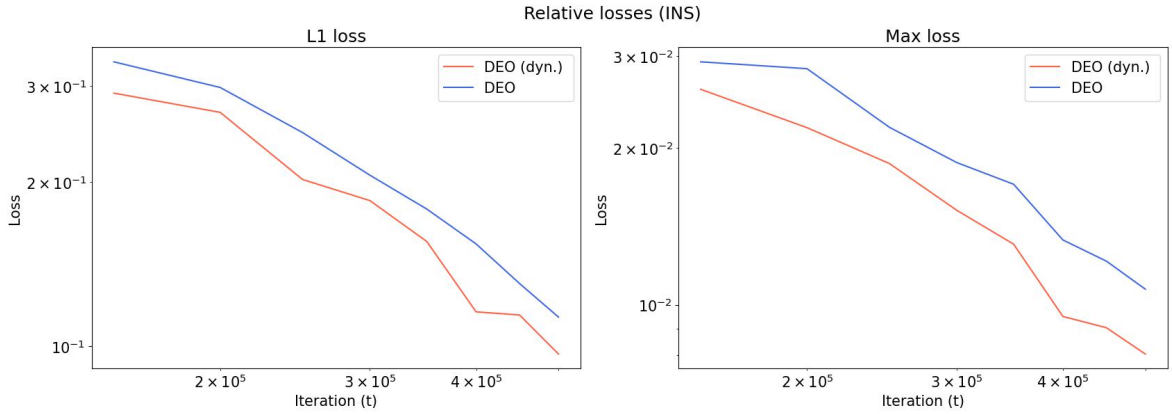

Figure 5.18: L1 and max relative losses for the dynamic and non-dynamic DEO structure MCMC (INS). $p(\text{REV}) = 0.7$ while $p(\text{MBR}) = 0.2$ for all the samplers. Averaged value over 5 structure MCMC runs containing 530 000 samples (using 100 000 as burn-in value) .

The averaged value of the relative losses is shown in Figure 5.18, while Table 5.6

shows the value of the losses after 500 000 iterations. Also in this case, the dynamic scheme performs better compared to the non-dynamic one. However, the relative losses show how fast the MCMC runs approach to each other, but no statement can be made about a correct convergence to the posterior distribution of each MCMC run.

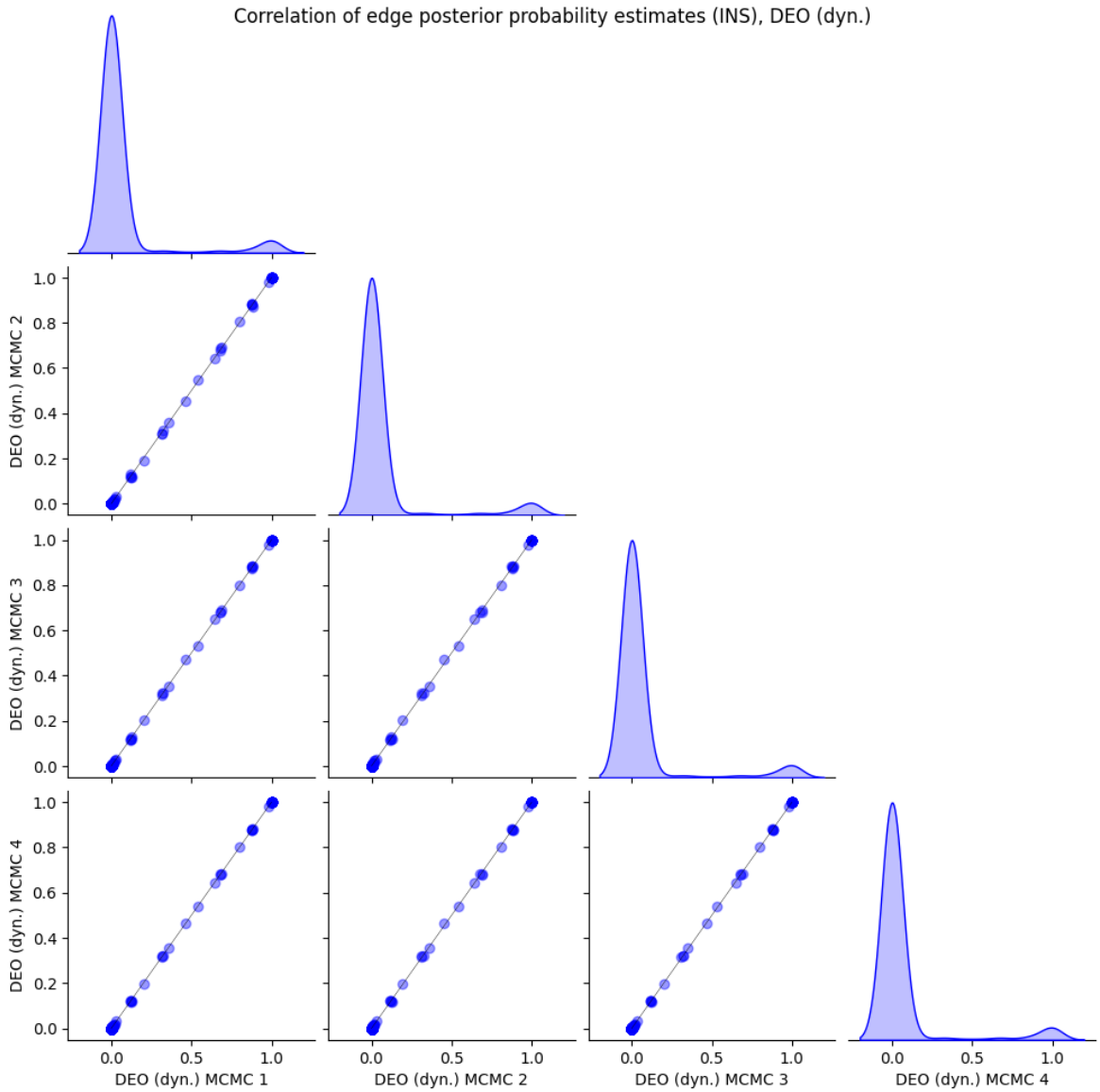

Figure 5.19: Correlation of the posterior probabilities of the edges among 4 dynamic DEO structure MCMC runs (INS) with $p(\text{REV}) = 0.7$ and $p(\text{MBR}) = 0.2$. The structure MCMC algorithms sampled 530 000 DAGs. The first 100 000 DAGs have been discarded. Each point of any sub-graph has coordinates equal to the probability of the presence of an edge going from X_i to X_j for the 2 compared algorithms.

Figure 5.19 shows the correlation of the posterior probability of the edges for 4 different dynamic DEO structure MCMC runs. The number of data points is 5 000 and

Table 5.6: Losses (INS) after 500 000 iterations using BIDA as reference. The burn-in value is 100 000.

Algorithm	L1 loss	Max-loss
DEO	0.113	0.011
DEO (dyn.)	**0.097**	**0.008**

the maximum in-degree is 3, consequently, the number of edges that have probability $1 \gg p(\text{edge}) \gg 0$ is small, since the strong likelihood term favors few high-scoring DAGs. The probability of the edges are heavily correlated for all the runs, giving evidence about the correct convergence of the MCMC runs to the target distribution.

Remarks regarding the convergence of the dynamic scheme

The dynamic tuning scheme outperforms the non-dynamic one when the posterior distribution is complex and multimodal (MUSH and INS). When DAGs are sampled from "simpler" posterior distributions (small number of variables N or weak likelihood term), the dynamic and non-dynamic schemes perform similarly. By looking at the behavior of the losses in the graphs shown above (log-scale is used for both axis), we can see that the losses of both algorithms are continuously dropping for all the MCMC runs. In addition, they do not converge to a fixed value greater than 0. This provides strong evidence in support of the claim that the dynamic sampler is not biased.

5.7.3 Round trips and rejection ratios comparisons

The averaged value of the number of round trips, over the different structure MCMC runs, has been calculated for all the algorithms, to compare the efficiency of the dynamic and non-dynamic communication schemes. The table below reports the averaged value of the round trips performed (after the first tuning phase) for each algorithm and, additionally, the value of the mean (over the different MCMC runs) of the mean of the rejection ratio among the chains of a single run (for practical purposes $\bar{r}$). Similarly, the mean (over the different MCMC runs) of the standard deviation of the rejection ratio among chains of a single run is reported. The DEO scheme outperforms the other schemes in terms of round trips performed and the dynamic tuning scheme outperforms the non-dynamic one. Additionally, the dynamic scheme ensures a smaller standard deviation for the values of the rejection ratio among chains. The averaged value of the mean of the rejection ratio among chains is $\mathbb{E}(\bar{r}) \approx 0.5$ for all the datasets, therefore, the estimation of the global communication barrier, obtained after the first tuning phase, is accurate in all the experiments. An integer value has to be selected for the

Table 5.7: Averaged value of the number of round trips performed by each algorithm. Additionally, the averaged value over the different runs of the mean and the standard deviation of the rejection ratio among chains are shown. The analyzed dataset is shown in the first column of the table.

Dataset	Algorithm	$\mathbb{E}$(Round trips)	$\mathbb{E}(\bar{r})$	$\mathbb{E}(\sigma(r))$
MUSH	DEO	390.5	0.493	0.026
	DEO (dyn.)	**402.75**	0.493	**0.001**
	SEO (dyn.)	378.5	0.493	**0.001**
	SRS (dyn.)	179.5	0.493	**0.001**
ZOO	DEO	2935.8	0.521	0.017
	DEO (dyn.)	**2974.2**	0.521	**0.001**
ASIA	DEO	11036.5	0.517	0.014
	DEO (dyn.)	**11217.25**	0.517	**0.001**
INS	DEO	121.2	0.487	0.041
	DEO (dyn.)	**123.4**	0.488	**0.004**

number of chains $C + 1$, hence, it is impossible to reach an exact value $\mathbb{E}(\bar{r}) = 0.5$ for the rejection ratio among chains.

5.8 Annealing schedule variation in dynamic tuning

In this section, the behavior of the temperatures is analyzed to illustrate how dynamic tuning affects the choice of the annealing schedule. In Figures 5.20, 5.21, and 5.22, the behavior of the temperatures during the two tuning phases are shown, respectively, for a structure MCMC run using ASIA, MUSH, and ZOO datasets. The algorithm used is the dynamic DEO structure MCMC.

5.8.1 First tuning phase

The first tuning phase is equivalent for both dynamic and not dynamic algorithms. The values of the temperatures are initialized using a linear scale. Therefore, $\beta_{c+1} = 1/C + \beta_c$. This is not an optimal annealing schedule. In the first tuning iterations, the change in value of the different temperatures is evident. After the temperature values converge, it is possible to stop the first tuning phase and obtain a suitable value for the optimal number of chains $\bar{C}+1$ using equation $\bar{C} = 2\bar{\Lambda}$ [10].

When the second tuning phase is not performed, sampling using the temperatures obtained in the first tuning phase may not lead to an efficient DAGs sampling, resulting in rejection rates among chains that have high variance, and that may lead to a biased approximation of the posterior distribution. Rejection rates can reach

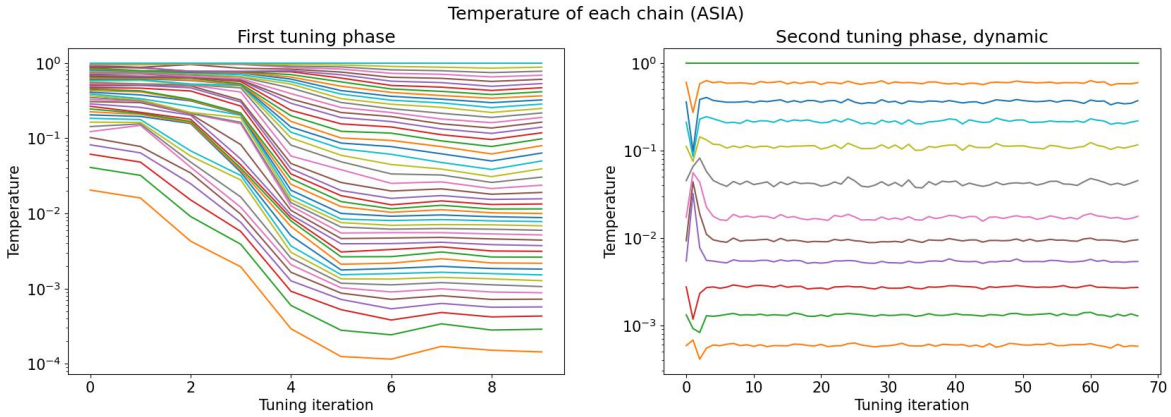

Figure 5.20: Temperature values for each chain during the first tuning phase (left) and during the second (dynamic) tuning phase (right) for ASIA. On the x-axis there are the the different tuning iterations and on the y-axis there are the values of the temperatures. Each line in the left graph represents the temperature variation of each of the 60 starting chains associated with a temperature β_c. Similarly, after an optimal number of chains $C+1=13$ is found, in the right graph, the behavior of the temperatures during the second tuning phase is shown. β_0 is omitted in both graphs to allow a correct visualization using log-scale for the y-axis.

values 1 or 0 resulting in complete lack of communication between chains at specific temperatures β_c and β_{c+1}. This is caused by the approximation made in Equation 4.18 to calculate the local communication barrier function [10]. When the GPU is available, it is possible to run more than 10 000 chains in parallel, therefore, a reasonable approximation can be achieved. If GPU is not available, it is not convenient to run, using CPU, the same amount of chains. It is indeed recommended to initialize fewer chains and perform a second tuning phase, where only the temperatures are tuned, to obtain a constant rejection ratio among chains (that should be $r^* \approx 0.5$).

The second tuning phase is recommended also when GPU is available, even if the approximation is still better than the one obtained by using a smaller starting number of chains. As already mentioned in Section 4.5.2, sampling from a sharp multimodal distribution requires an extremely accurate value of the temperatures.

5.8.2 Second tuning phase

In the second tuning phase the values of the temperatures are updated after 3 000 iterations. During the first tuning steps, the values for the temperatures change drastically. Then, they stabilize and oscillate around values that are similar to the starting values. The change during the first iterations was expected, as the DAGs selected as starting point of the second tuning phases are the $C+1$ current DAGs that were belonging to the warmer chains after the first tuning phase (the DAGs that

were in the colder chains have been discarded). Therefore, the rejection rates change during the first steps of the second tuning phase since the DAGs do not "represent well" the target distribution of each chain.

However, it is not trivial that the values of the temperatures do not oscillate significantly around some fixed values, even when the chains are exploring different modes. This result shows how dynamic tuning does not strongly affect the sampling process. This oscillation may differ in intensity depending on the number of iterations performed during each tuning step of the second tuning phase.

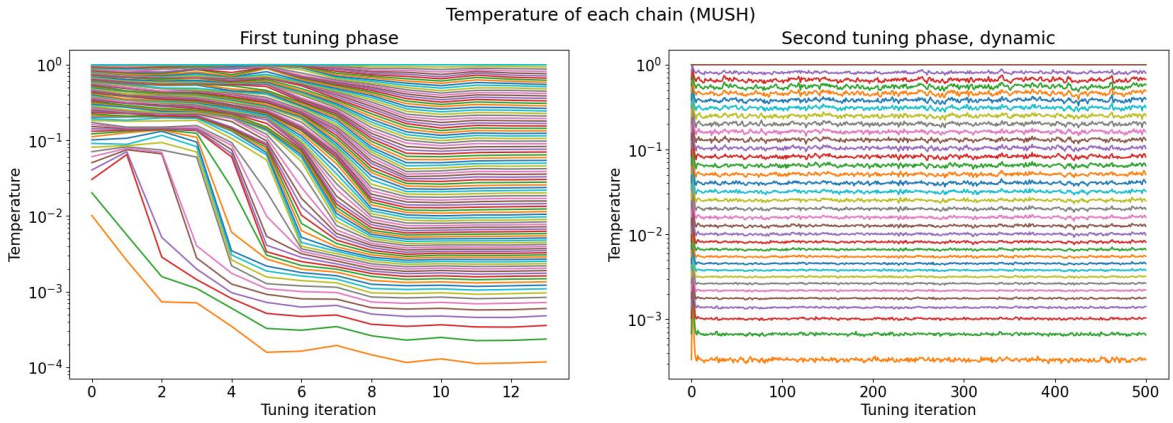

Figure 5.21: Temperature values for each chain during the first tuning phase (left) and during the second (dynamic) tuning phase (right) for MUSH. On the x-axis there are the different tuning iterations and on the y-axis there are the values of the temperature. β_0 is omitted in both graphs to allow a correct visualization using log-scale for the y-axis.

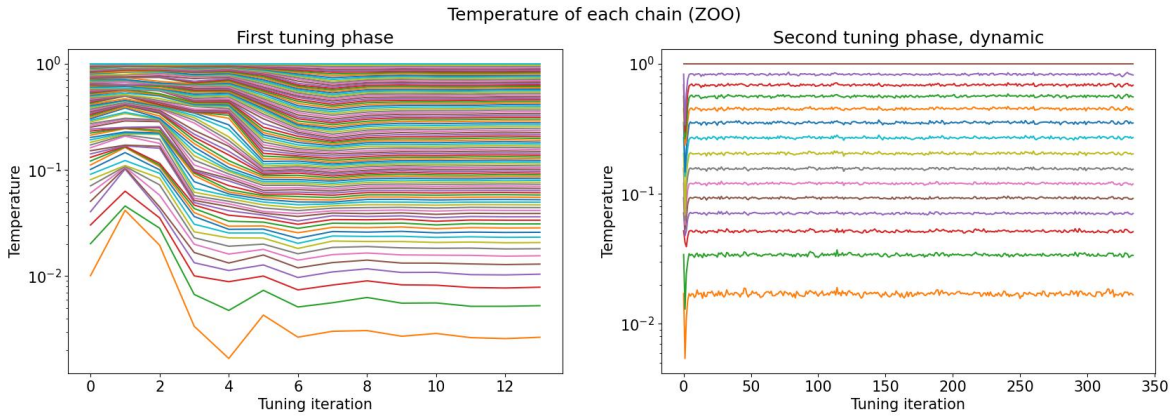

Figure 5.22: Temperature values for each chain during the first tuning phase (left) and during the second (dynamic) tuning phase (right) for ZOO. On the x-axis there are the different tuning iterations and on the y-axis there are the values of the temperature. β_0 is omitted in both graphs to allow a correct visualization using log-scale for the y-axis.

6. Conclusions

Approximating the posterior distribution of DAGs and finding the conditional relationships among variables starting from data is a difficult task. As already mentioned in Chapter 3, the space of DAGs grows super-exponentially with the number of nodes N of the network. The more data points are added, the sharper the posterior distribution is, with high-scoring DAGs separated by low-probability regions. The structure MCMC algorithm samples DAGs with high autocorrelation, thus, in practice, it is not able to cross the low-probability regions of the posterior distribution. Hence, an optimal tuning of the hyperparameters of the structure MCMC, combined with the implementation of a parallel tempering scheme, help to overcome this limitation.

The experiments shown in the previous chapter have multiple scopes. The performance of the different parallel tempering schemes were compared when combined with the structure MCMC algorithm. Furthermore, the properties of each structure MCMC move were analyzed in depth, in order to describe an optimal and flexible sampler that generalizes well and works efficiently for all the examined networks, without requiring further tuning for every experiment.

We provide surprising evidence that assigning a higher probability than 1/15 to REV and MBR moves can be beneficial. Su and Borsuk [8] state that the optimal probability of performing a REV move and a MBR move, for the structure MCMC, is 1/15. These values were computed using a reference distribution that could potentially be biased, since it was obtained by using a structure MCMC, initialized at a high-scoring DAG. The MC^3 move is known to be less computationally demanding compared to the REV and the MBR moves, but it proposes DAGs that differ only by one edge from the current DAG. A sampler that performs this move with a high probability can easily get stuck near local maxima, generating a biased posterior distribution.

The REV move and the MBR move perform better compared to the MC^3 move and are a useful tool to cross low-probability regions. The REV move performs better than the MBR move since it can reverse an edge, add, and remove edges in a single

move, while the MBR move can only add and remove edges. Hyperparameter tuning showed that a sampler should perform REV moves with a higher probability compared to MBR moves. The probabilities for each move, determined through hyperparameter tuning in experiments involving the ASIA dataset, generalize well to the other analyzed networks and significantly enhance the performance of the structure MCMC algorithm.

However, even an optimally tuned structure MCMC may get stuck near local modes. We provided strong evidence that the structure MCMC, provided with a parallel tempering algorithm, correctly explores the parameter space. The comparison among different parallel tempering schemes showed that, according to the theory, the DEO communication scheme, presented in Section 4.2.2, outperforms the other schemes.

A novel tuning routine for the temperatures, divided into two phases, has been suggested. It ensures good communication between chains, using less computational resources compared to the training routine suggested by Syed et al. [10], which could be only performed when GPU is available (since it requires an extremely high number of chains that are executed in parallel, to obtain an optimal value of the annealing schedule).

In addition, a dynamic parallel tempering scheme has been proposed to further increase communication between chains, especially when the hyperparameters of the samplers are not initialized correctly. Dynamic tuning is enabled by not stopping the second tuning phase. In the experiments, we kept the number of iterations of each tuning step of the second tuning phase fixed, to obtain a flexible sampler that may update the temperatures significantly with few iterations. This approach ensures a fast escape for the sampler from possible local modes of the posterior distribution. Even if there is no theory that proves that the samples obtained using the dynamic algorithm converge to the stationary distribution, the experiments show that the dynamic scheme outperforms the standard scheme in challenging situations, when the number of nodes increases and the posterior distribution is multimodal. These results provide strong evidence regarding its unbiasedness. This novel dynamic parallel tempering scheme can also be easily implemented to sample from any type of distribution, outside of the Bayesian networks setting.

It remains open whether it is possible to overcome the limits of the current structure MCMC moves. The need of a parallel tempering scheme for the structure MCMC underlines how poorly the algorithm performs when the posterior distribution presents

low-probability regions among modes. Sampled DAGs present high auto-correlation. This feature leads to benefits and disadvantages. When the sampler reaches the main mode, only high-scoring DAGs are proposed, leaving out low-scoring DAGs and reducing the computation. However, it can make model-averaging biased, when the posterior distribution is multimodal. New efforts could be made to find new structure MCMC moves that could increase the size of the neighborhood of any DAG using fewer computational resources. New moves should also ensure a high acceptance ratio of the proposed state, to efficiently explore the posterior distribution.

Bibliography

[1] Nir Friedman and Daphne Koller. Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning*, 50:95–125, Jan 2003.

[2] David Heckerman, Christopher Meek, and Gregory Cooper. *A Bayesian Approach to Causal Discovery*, pages 1–28. Springer, 2006.

[3] David Madigan, Jeremy York, and Denis Allard. Bayesian Graphical Models for Discrete Data. *International Statistical Review / Revue Internationale de Statistique*, 63(2):215–232, Aug 1995.

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

[5] Wilfred K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, Apr 1970.

[6] Paolo Giudici and Robert Castelo. Improving Markov Chain Monte Carlo Model Search for Data Mining. *Machine Learning*, 50:127–158, Jan 2003.

[7] Marco Grzegorczyk and Dirk Husmeier. Improving the Structure MCMC Sampler for Bayesian Networks by Introducing a New Edge Reversal Move. *Machine Learning*, 71(2-3):265–305, Apr 2008.

[8] Chengwei Su and Mark E Borsuk. Improving Structure MCMC for Bayesian Networks through Markov Blanket Resampling. *Journal of Machine Learning Research*, 17(1):4042–4061, Jan 2016.

[9] Charles J. Geyer. Markov Chain Monte Carlo Maximum Likelihood. *University Digital Conservancy*, Aug 1991.

[10] Saifuddin Syed, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Non-Reversible Parallel Tempering: a Scalable Highly Parallel MCMC Scheme. *Journal of the Royal Statistical Society (Series B)*, 84:321–350, May 2021.

[11] Tsuneyasu Okabe, Masaaki Kawata, Yuko Okamoto, and Masuhiro Mikami. Replica-Exchange Monte Carlo Method for the Isobaric–Isothermal Ensemble. *Chemical Physics Letters*, 335(5-6):435–439, Mar 2001.

[12] Yves F. Atchadé, Gareth O. Roberts, and Jeffrey S. Rosenthal. Towards Optimal Scaling of Metropolis-Coupled Markov Chain Monte Carlo. *Statistics and Computing*, 21(4):555–568, Jul 2010.

[13] Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, Jan 2014.

[14] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, Nov 1984.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1993.

[16] Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2004.

[17] Steffen L. Lauritzen and David J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, Jan 1988.

[18] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[19] Thomas Verma and Judea Pearl. Equivalence and Synthesis of Causal Models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pages 255–270. Elsevier Science Inc., Jul 1990.

[20] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive Computation and Machine Learning. MIT Press, 2009.

[21] Ralf Eggeling, Jussi Viinikka, Aleksis Vuoksenmaa, and Mikko Koivisto. On Structure Priors for Learning Bayesian Networks. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1687–1695, Apr 2019.

[22] Dan Geiger and David Heckerman. Learning Gaussian Networks. In *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*, pages 235–243, Jul 1994.

[23] Gregory F. Cooper and Edward Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309–347, Oct 1992.

[24] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243, Sep 1995.

[25] Joe Suzuki. A Theoretical Analysis of the BDeu Scores in Bayesian Network Structure Learning. *Behaviormetrika*, 44(1):97–116, Nov 2016.

[26] Marco Scutari. An Empirical-Bayes Score for Discrete Bayesian Networks. *Journal of Machine Learning Research*, 52:438–448, May 2016.

[27] Robert W. Robinson. Counting Unlabeled Acyclic Digraphs. In Charles H. C. Little, editor, *Combinatorial Mathematics V*, pages 28–43. Springer, 1977.

[28] Topi Talvitie, Aleksis Vuoksenmaa, and Mikko Koivisto. Exact Sampling of Directed Acyclic Graphs from Modular Distributions. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 965–974, Jul 2020.

[29] Arthur B. Kahn. Topological Sorting of Large Networks. *Communications of the ACM*, 5(11):558–562, Nov 1962.

[30] Juha Harviainen. Personal Communication, Dec 2023.

[31] Luke Tierney. Markov Chains for Exploring Posterior Distributions. *Annals of Statistics*, 22(4):1701–1728, Dec 1994.

[32] Martin Lingenheil, Robert Denschlag, Gerald Mathias, and Paul Tavan. Efficiency of Exchange Schemes in Replica Exchange. *Chemical Physics Letters*, 478(1-3):80–84, Aug 2009.

[33] Cristian Predescu, Mihaela Predescu, and Cristian V. Ciobanu. The Incomplete Beta Function Law for Parallel Tempering Sampling of Classical Canonical Systems. *The Journal of Chemical Physics*, 120(9):4119–4128, Mar 2004.

[34] Fred N. Fritsch and Robert E. Carlson. Monotone Piecewise Cubic Interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, Apr 1980.

[35] Walter Nadler and Ulrich H. E. Hansmann. Dynamics and Optimal Number of Replicas in Parallel Tempering Simulations. *Phisical Review E*, 76:065701, Dec 2007.

[36] Aminata Kone and David A. Kofke. Selection of Temperature Intervals for Parallel-Tempering Simulations. *Journal of Chemical Physics*, 122(20):206101, May 2005.

[37] Helmut G. Katzgraber, Simon Trebst, David A. Huse, and Matthias Troyer. Feedback-Optimized Parallel Tempering Monte Carlo. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(03):P03018, Mar 2006.

[38] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Brooks/Cole, Cengage Learning, 9th edition, 2011. Chapter on Bisection Method.

[39] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[40] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array Programming with NumPy. *Nature*, 585(7825):357–362, Sep 2020.

[41] James Cussens. Bayesian Network Learning with Cutting Planes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 153–160, 2011.

[42] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive Probabilistic Networks with Hidden Variables. *Machine Learning*, 29(2–3):213–244, Nov 1997.

[43] Mushroom. UCI Machine Learning Repository, 1987. DOI: https://doi.org/10.24432/C5959T.

[44] Richard Forsyth. Zoo. UCI Machine Learning Repository, 1990. DOI: https://doi.org/10.24432/C5R59V.

[45] Johan Pensar, Topi Talvitie, Antti Hyttinen, and Mikko Koivisto. A Bayesian Approach for Estimating Causal Effects from Observational Data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5395–5402, Apr 2020.

[46] Jin Tian and Ru He. Computing Posterior Probabilities of Structural Features in Bayesian Networks. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 538–547, 2009.

[47] Jack Kuipers and Giusi Moffa. Uniform Random Generation of Large Acyclic Digraphs. *Statistics and Computing*, 25(2):227–242, Nov 2013.

[48] Jack Kuipers and Giusi Moffa. Partition MCMC for Inference on Acyclic Digraphs. *Journal of the American Statistical Association*, 112(517):282–299, Jan 2017.

[49] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, Jan 2001.

[50] Marloes H. Maathuis, Markus Kalisch, and Peter Bühlmann. Estimating High-Dimensional Intervention Effects from Observational Data. *The Annals of Statistics*, 37(6A):3133–3164, Dec 2009.

[51] James R Norris. *Markov Chains*. Cambridge University Press, Cambridge, 1998.

Appendix A. Stationary distribution of the Lazy MC^3

Let us first clarify the notation. The DAG M is the current state of the Markov chain, while M' is the proposed state. The set $\text{nbd}(M^*)$ contains all neighboring DAGs of M^* that can be obtained by adding, removing, or reversing one edge. The value of $|\text{nbd}(M^*)|$ is bounded by $N(N-1)$ where N is the number of nodes. There are three different ways to reach the DAG M^* after an iteration:

- **First case**. The DAG M^* can be reached by moving from one of the neighbors.
- **Second case**. The DAG M^* can be reached by being in the state $M = M^*$ and the proposal $M' \neq M$ is rejected.
- **Third case**. The DAG M^* can be reached by setting $M = M' = M^*$, in other words, the proposed DAG is the same as the current DAG.

We will expand these three terms separately. The event acc. expresses the condition "M' is accepted" (while rej. expresses the condition "M' is rejected"). The first case occurs with probability

$$\begin{aligned} &p(M \in \text{nbd}(M^*), M' = M^*, \text{acc.}) \\ &= \sum_{M^{\pm} \in \text{nbd}(M^*)} \frac{1}{N(N-1)} p(M^{\pm}) \min\left\{1, \frac{p(M^*)}{p(M^{\pm})}\right\} \\ &= \frac{1}{N(N-1)} \sum_{M^{\pm} \in \text{nbd}(M^*)} \min\left\{p(M^{\pm}), p(M^*)\right\} \quad , \end{aligned} \tag{A.1}$$

where $M^{\pm}$ denotes a neighbor of M^*. The second case occurs with probability

$$\begin{aligned} &p(M = M^*, M' \in \text{nbd}(M^*), \text{rej.}) \\ &= \sum_{M^{\pm} \in \text{nbd}(M^*)} p(M^*) \frac{1}{N(N-1)} \left(1 - \min\left\{1, \frac{p(M^{\pm})}{p(M^*)}\right\}\right) \\ &= p(M^*) \frac{|\text{nbd}(M^*)|}{N(N-1)} - \frac{1}{N(N-1)} \sum_{M^{\pm} \in \text{nbd}(M^*)} \min\left\{p(M^*), p(M^{\pm})\right\} \quad , \end{aligned} \tag{A.2}$$

and the third case occurs with probability

$$p(M = M' = M^*) = p(M^*)\left(1 - \frac{|\text{nbd}(M^*)|}{N(N-1)}\right) \quad . \tag{A.3}$$

The total probability of the three cases is

$$\begin{aligned}
&\frac{1}{N(N-1)} \sum_{M^{\pm} \in \text{nbd}(M^*)} \min\left\{p(M^{\pm}), p(M^*)\right\}(1-1) \\
&+ p(M^*)\frac{|\text{nbd}(M^*)|}{N(N-1)}(1-1) + p(M^*) \\
&= p(M^*) \quad .
\end{aligned} \tag{A.4}$$

Therefore, the transition kernel of the Lazy MC^3 sampler leaves the distribution $p(M^*)$ invariant. Now we want to prove the ergodicity of the Lazy MC^3 sampler. There are three conditions [51] that must be satisfied:

- **Irreducibility**. When no edge constraints are given to the network, the posterior distribution assigns a positive probability (lower than 1) to all the DAGs that satisfy the maximum in-degree condition. Therefore, we can transform any DAG to an empty DAG by performing a sequence of edge removals. Then, any valid DAG can be obtained by performing a series of edge additions. This series of operations can occur with positive probability. Therefore, the Lazy MC^3 sampler satisfies the irreducibility condition, since every DAG is reachable from any other DAG in a finite number of exploration steps.

- **Aperiodicity**. Since we already proved that the Lazy MC^3 sampler satisfies the irreducibility condition, to show the aperiodicity of the Markov chain, it is sufficient to show that there is one state that is aperiodic. Let as assume that the current state is the empty DAG. Consequently, any pair of nodes u and v are not connected and no cycles can be formed when an edge between these nodes is added to the DAG. Two cycles are obtained by

  1. adding the edge (u, v) and removing the edge (u, v);
  2. adding the edge (u, v), reversing the edge (u, v), and removing the edge (v, u).

  These cycles occur with positive probability and are performed in 2 and 3 steps respectively. The greatest common divisor of 2 and 3 is 1, and thus the Lazy MC^3 satisfies the aperiodicity condition.

- **Positive recurrence**. We know that the space of DAGs given the number of nodes N is finite (Equation 3.1) and that the Lazy MC^3 sampler satisfies the irreducibility condition. Therefore, the Lazy MC^3 sampler returns to any DAG in a fixed expected number of iterations.

Since the Lazy MC^3 sampler is ergodic and leaves the distribution $p(M^*)$ invariant, we can now conclude that the Lazy MC^3 converges to the stationary distribution $p(M^*)$.

Appendix B. Round trips analysis for the SRS scheme

The notation that has been used for the proof is similar to the one of Appendix B of the paper presented by Syed et al. [10]. $\mathcal{T}_\uparrow$ and $\mathcal{T}_\downarrow$ are the quantities that are used to calculate the round trip rate. $\mathcal{T}_\uparrow$ represents the number of iterations performed by the analyzed sampling machine to reach the temperature index C, and $\mathcal{T}_\downarrow$ represents the number of iterations performed by the sampling machine to reach the temperature index 0. As already explained in Chapter 4, when $c = 0$ the machine samples from the prior distribution and when $c = C$ the machine samples from the posterior distribution. We can now define

$$A_\bullet^c = \mathbb{E}_{\text{SRS}}(\mathcal{T}_\bullet \mid c(t=0) = c) \quad , \tag{B.1}$$

where $\bullet = \{\uparrow, \downarrow\}$. In words, $A_\bullet^c$ is the expected value of $\mathcal{T}_\bullet$ knowing that the initial index of the temperature of the sampling machine is equal to c (we know that $0 \leq c \leq C$). Therefore, the expected number of iterations to perform a round trip is

$$\mathbb{E}_{\text{SRS}}(\mathcal{T}) = A_\uparrow^0 + A_\downarrow^C \quad . \tag{B.2}$$

Wee denote as r_c the rejection ratio between chains c and $c+1$ and as s_c the acceptance ratio between chain c and chain $c+1$. The Markov property ensures that, for every c and for every $\bullet \in \{\uparrow, \downarrow\}$

$$A_\bullet^c = \frac{1}{C} s_c (A_\bullet^{c+1} + 1) + \frac{1}{C} s_{c-1} (A_\bullet^{c-1} + 1) + \left(1 - \frac{2}{C}\right)(A_\bullet^c + 1) \quad . \tag{B.3}$$

For $i = \{1, \ldots, C\}$, Equation B.3 is equivalent to

$$-C = s_c B_\bullet^{c+1} + s_{c-1} B_\bullet^c \quad , \tag{B.4}$$

where $B_\bullet^c = A_\bullet^c - A_\bullet^{c-1}$. Using recursion, Equation B.4 can be written as

$$s_{c-1} B_\bullet^c = s_0 B_\bullet^1 - C(c-1) \quad , \tag{B.5}$$

or equivalently as

$$s_{c-1} B_\bullet^c = s_{C-1} B_\bullet^C + C(C-c) \quad . \tag{B.6}$$

As already expressed by Syed et al. [10], let us now analyse the $\uparrow$ and $\downarrow$ cases separately. If $c(t=0)=0$, then the probability of obtaining $c(t=1)=1$ is equal to $\frac{1}{C}s_0$. Hence,

$$A_{\uparrow}^{0} = \frac{1}{C}s_0(A_{\uparrow}^{1}+1) + \left(1-\frac{1}{C}s_0\right)(A_{\uparrow}^{0}+1) \quad , \tag{B.7}$$

which can be reduced to

$$s_0 B_{\uparrow}^{1} = -C \quad . \tag{B.8}$$

Equation B.8 can be integrated with Equation B.5 to get the relation

$$s_{c-1} B_{\uparrow}^{c} = -Cc \quad . \tag{B.9}$$

From Equation B.9, by recursion, we obtain

$$A_{\uparrow}^{0} = A_{\uparrow}^{c} + \sum_{k=1}^{c} \frac{Ck}{s_{k-1}} \quad , \tag{B.10}$$

or, equivalently,

$$A_{\uparrow}^{0} = \sum_{k=1}^{C} \frac{Ck}{s_{k-1}} \quad , \tag{B.11}$$

since $A_{\uparrow}^{C}=0$. The same considerations of Equation B.7 can be made to obtain $A_{\downarrow}^{C}$. Therefore,

$$A_{\downarrow}^{C} = \frac{1}{C}s_{C-1}(A_{\downarrow}^{C-1}+1) + \left(1-\frac{1}{C}s_{C-1}\right)(A_{\downarrow}^{C}+1) \quad . \tag{B.12}$$

Equation B.12 implies

$$s_{C-1} B_{\downarrow}^{C} = C \quad . \tag{B.13}$$

By integrating Equations B.6 and B.13 we obtain

$$s_{c-1} B_{\downarrow}^{c} = C(C-c+1) \quad , \tag{B.14}$$

and, by recursion we get

$$A_{\downarrow}^{C} = A_{\downarrow}^{c} + \sum_{k=c+1}^{C} \frac{C(C-k+1)}{s_{k-1}} \quad . \tag{B.15}$$

For $c = C$, Equation B.15 becomes

$$A_{\downarrow}^{C} = \sum_{k=1}^{C} \frac{C(C-k+1)}{s_{k-1}} \quad , \tag{B.16}$$

since $A_{\downarrow}^{0} = 0$. By integrating Equations B.2, B.11, and B.16, we get the formula of the expected number of iterations per round trip of the SRS scheme

$$\begin{aligned}
\mathbb{E}_{\mathrm{SRS}}(\mathcal{T}) &= \sum_{k=1}^{C} \frac{Ck}{s_{k-1}} + \sum_{k=1}^{C} \frac{C(C-k+1)}{s_{k-1}} \\
&= \sum_{k=1}^{C} \frac{C(C+1)}{s_{k-1}} \\
&= C^2(C+1) + C(C+1) \sum_{k=1}^{C} \frac{r_{k-1}}{s_{k-1}} \\
&= C^2(C+1) + C(C+1) \sum_{k=0}^{C-1} \frac{r_k}{s_k} \quad .
\end{aligned} \tag{B.17}$$