Bayesian vs. Constraint-Based Causal Discovery Algorithms: A Monte Carlo Simulation Study

Niklas Lienau[1]

[1]Goethe University Frankfurt

March 20, 2025

Abstract

This paper compares bayesian and constraint-based causal discovery methods in recovering the underlying structure of Bayesian networks. I conduct a Monte Carlo simulation study evaluating the order-MCMC algorithm and the PC algorithm across varying sample sizes, network sizes, sparsity levels, and edge strengths. Performance is assessed using the Structural Hamming Distance (SHD), the True Positive Rate (TPR) and the False Positive Rate (FPR). My results show that both approaches perform similarly in recovering undirected network structures, though Bayesian methods offer additional advantages such as uncertainty quantification and full posterior inference over directed graphs.

1 Background

1.1 Graphical Models: Bayesian Networks and DAG's

A **Directed Acyclical Graph** (DAG) is a graphical model which visualizes dependence relationships in multivariate distributions. Any Directed Acyclical Graph (DAG) consists of nodes representing random variables and directed edges that encode dependencies between them. The key constraint is acyclicity, ensuring that the graph produces no cycles. **Bayesian Networks** (BN) [Pearl, 1988] $\mathcal{B} = (\mathcal{G}, \theta)$ extend this setting to a fully specified probabilistic model by equipping the DAG $\mathcal{G}$ with a set of parameters θ so that the two jointly define the conditional distributions in the system encoded by the DAG. The **Causal Markov Assumption** states that in the underlying probability distribution of the random variables in the system, every variable is independent of its non-descendants conditional on its parents as specified by the DAG. Conceptually this assumes the DAG is correctly specified and indeed visualizes the true network structure. Using this assumption, the joint probability function of the n random variables $X_1, ..., X_n$ can be factorized as:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Parents}(X_i)) \tag{1}$$

The individual conditional distributions in equation 1 are then parameterized by θ.
For illustration consider the case where the data follows a multivariate normal $(X_1, ..., X_n) \sim N(\mu, \Sigma)$. A Gausian Bayesian Network (GBN) would include a DAG $\mathcal{G}$ and a set of parameters $\theta = \{\beta, \sigma, \alpha\}$ so that $P(X_1, ..., X_n)$ gets factorized according to equation 1 and the individual conditional distributions are given by:

$$X_i \mid \text{Parents}(X_i) \sim \mathcal{N}\left(\sum_{X_j \in \text{Parents}(X_i)} \beta_{ij} X_j + \alpha_i, \sigma_i^2\right) \tag{2}$$

The parameter set consists in this example of linear combination coefficients for the mean β, the intercepts α and the conditional variances σ.
Bayesian networks and DAG's have found prominent use across science and business applications. DAG's, which are purely graphical models, help visualize scientific understanding of complex systems and networks. In Causal Inference they can be used to determine which causal target objects can be identified given the DAG and the data (for example with the help of Judea Pearls "do calculus"). Bayesian networks, which are complete probabilistic models, have been used for inference

and as decision making tools across many scientific disciplines e.g for medical diagnosis or fraud detection.
While the usefulness of these graphical models is undebated, in practice it is often difficult to come up with a DAG $\mathcal{G}$. They are either manually specified from an expert with domain expertise or learned from data by an algorithm. The next section introduces the most prominent and widely used structure learning algorithms.

1.2 Structure Learning & Causal Discovery

Sometimes it is infeasible to manually specify a DAG from expert knowledge. This especially arises in very high dimensional networks where the number of potential dependencies grows too large and the task of drawing a DAG becomes to complex. The goal then is to obtain algorithms that analyze the data and return the underlying data generating DAG. These algorithms are sometimes called **structure learning algorithms** and sometimes **causal discovery algorithms** depending on wether the resulting DAG is interpreted in a causal or a purely statistical way. In the following I will use these terms interchangeably. Causal discovery algorithms can be divided into two families: **Constrained based** and **search and score based** algorithms.

- **Constrained based algorithms** rely on conditional independence tests and implement the idea, that if variables are independent in the data generating DAG, they should also be statistically independent in the sample data. The most prominent constrained based algorithm is the **PC algorithm** of [Spirtes et al., 2001] for which [Kalisch and Bühlman, 2007] recently developed consistency results and a popular R package called "pcalg". Because the conditioning set for every given node is of size 2^{n-2} and grows large quickly, the algorithm reduces computational cost by starting from a fully connected (but undirected) graph and then performing statistical correlation tests in the order of increasing complexity. Finally, only edges that are not deleted in any iteration of all the conditional independence tests are kept and form the final **skelaton** (the DAG with undirected edges). This stems from the fact that statistical correlation tests are direction blind. However, in a last step some edges can be directed by exploiting the presence of colliders which generate correlation patterns that identify so called **v-structures** in the graph. Finally, the algorithm returns a partially directed graph (CPDAG) which represents the **Markov equivalence class**, namely the set of DAG's consistent with the statistical dependence relationships observed in the data. In this sense constrained based approaches can only perform partial identification as they do not identify a single DAG but a set consisting of multiple graphs all consistent with the observed data.

- The other prominent class of approaches to learn the graphical structure from data are called **search and score based algorithms**. The main idea is to assign each graph a score and search the DAG space for the highest-scoring candidate. While there are also non mcmc-based approaches like greedy search algorithms, this paper will focus on the bayesian approaches where a prior is defined over graphs and bayesian inference is used to obtain a full posterior distribution over DAG's :

$$P(G \mid D) \propto P(D \mid G)P(G) \tag{3}$$

  where $P(G)$ is a discrete prior distribution over all graphs and $P(D \mid G)$ is the marginal likelihood :

$$P(D \mid G) = \int P(D \mid \theta, G)P(\theta \mid G)\, d\theta \tag{4}$$

  Equation 4 integrates over the parameter priors $P(\theta \mid G)$ to compute the evidence for a DAG given the data. In total, bayesian inference requires two priors in this set up: The prior across graphs $P(G)$ and the prior over parameters within a given graph $P(\theta \mid G)$. $P(G)$ is often chosen to be a uniform distribution. When working with continuous data that can be assumed to follow a multivariate normal, $P(\theta \mid G)$ is often specified as a Normal-wishart prior so that due to conjugacy the integral in equation 4 has a closed-form solution. Given these prior choices the resulting score, namely the final posterior probability $P(G \mid D)$, is called the **Bayesian Gaussian equivalent** (BGe) score. On top of conjugacy the BGe score fulfills certain conditions called **structure and parameter modularity** [Geiger and Heckerman, 2002] which allow a useful decomposition of equation 3 :

$$P(G \mid D) \propto P(D \mid G)P(G) = \prod_{i=1}^{n} M(X_i, Parents(X_i) \mid D) \tag{5}$$

This decomposition is computationally powerful because it shows that the total score is a product of terms for each node in the graph. The term for each node is a function $M(\cdot)$ that only depends on its parents, so that when a local change is made to a graph, one does not have to evaluate the entire network again, but can just evaluate locally at the nodes that were directly affected.
The most basic **markov chain monte carlo** (MCMC) algorithm to sample from the posterior distribution in 3 is called **structure MCMC**[Madigan et al., 1995]. The idea is to set up a markov chain whose stationary distribution is the target posterior. This is achieved by mimicking a classical **Metropolis Hastings** (MH) approach in the DAG space. The proposal mechanism picks a random edge and either deletes or adds it (depending on wether it initially existed). The acceptance ratio is computed so that the markov chain converges towards its target distribution. The structure MCMC algorithm has computational problems stemming from the special nature of its state space of graphical DAG objects. Firstly, the state space grows super exponentially with the number of nodes (variables) in the data. Unlike a normal parameter space with a continuous likelihood function, the DAG state space is discrete and small changes induced by the proposal mechanism can yield large likelihood jumps. Deleting or adding an edge can change the entire dependency structure of the network. This in turn results in slow convergence, bad mixing and the inability to fully explore the sometimes huge space of potential DAG's. To overcome these difficulties and provide a computational superior MCMC alternative, [Friedman and Koller, 2003] developed a sampling algorithm called **order-MCMC**, that runs over the space of topological orders, which is of magnitudes smaller than the space of DAGS. These topological orders are node orderings, where each of the orders represent a set of DAG's consistent with the ordering. Since the posterior over topological orderings is not of interest, one can recover the posterior over DAG's by sampling the highest-scoring DAG within each ordering with greedy search algorithms.
Since then many advancements have been made, including approaches that combine constrained-based and score-based algorithms. [Kuipers et al., 2022] for example identify the Markov equivalence class with a PC algorithm in the first step and then run order-MCMC over this reduced search space.

In theory the bayesian approach holds some meaningful advantages over constrained based algorithms when performing causal discovery. Through the use of priors one can incorporate expert knowledge and choose a hybrid approach between fully data driven DAG selection and complete manual specification. Additionaly the bayesian approach yields full uncertainty quantification by providing a posterior distribution over DAG's. While the learned Markov equivalence class is often practically not directly useful, the posterior can be used for **bayesian model averaging** (BMA), e.g estimating the probability of structural features of interest like the probability of an edge going from random variable X_i to X_j:

$$P(X_i \to X_j \mid D) = \sum_{G \in \mathcal{G}} \mathbf{1}(X_i \to X_j \in G) P(G \mid D) \tag{6}$$

2 Simulation Design

To compare the practical performance of bayesian causal discovery and constrained based methods I employ a Monte Carlo simulation study. Specifically I compare the order-MCMC algorithm with the PC algorithm across different simulation design settings, varying the network size (n), the sample size (N), the network density (d) and the edge strength (θ).

2.1 DAG & Data Simulation

In the first step, a design setting (network size n, network density d and edge strength θ) is fixed and a random DAG is sampled that is consistent with the chosen parameter specification for network size, network sparsity and edge strength. I iterate through many design settings generating networks with size $n \in \{5, 15, 25\}$ being either dense or sparse $d \in \{dense, sparse\}$ and having either strong or weak edge strength $\theta \in \{weak, strong\}$. The DAG is parameterized and extended to a fully specified GBN with constant conditional variance $\sigma^2 = 1$ and no intercept. The edge strength relates to the conditional mean parameters β that specify how strong the variables in the network are related. Specifically $\theta \in \{weak, strong\}$ determines wether the β coefficients on the edges in the network are drawn from a uniform distribution with interval bounds $weak = [0.1, 1]$ or

$strong = [1, 3]$. Finally for each Gausian Bayesian Network generated from a given design setting, I simulate data of sample sizes $N \in \{10, 50, 100, 1000\}$.

2.2 Evaluation Metrics

The PC and the order-MCMC algorithm return two fundamentally different objects. The PC algorithm learns the Markov equivalence class, namely the set of DAGs that generate the same conditional independence relationships in the data and which represents the partially directed DAG (CPDAG). The order-MCMC on the other hand, returns a full posterior probability distribution over all (fully directed) DAG's. Therefore it is not trivial to come up with a proper evaluation metric on which to compare the performance of these two algorithms. In this simulation study, I extract the **Maximum A Posteriori** (MAP) DAG from the order-MCMC output and replace all directed edges with undirected edges. This procedure yields a proper comparison candidate to the CPDAG from the PC algorithm and allows for a comparison study that studies the performance of recovering the undirected structure in the network. It has to be said however, that this approach completely disregards all the benefits that the bayesian approach has in the first place like complete uncertainty quantification and the ability to identify directionality. In this sense, this choice allows for a feasible comparison but the result can not be interpreted as a finding of which algorithm performs "better" in a very general sense as we restrict our definition of performance to a domain where the benefits of the bayesian approach are not even considered.

To study which approach is better in learning undirected network structure from data I use the following evaluation metrics: The **Structural Hamming Distance** (SHD),the **True Positive Ratio** (TPR) and the **False Positive Ratio** (FPR). The SHD measures the number of edge changes needed to transform one graph into another. These changes include edge insertions (missing edges in the learned graph that exist in the true DAG), edge deletions (extra edges in the learned graph that do not exist in the true DAG), and edge flips (incorrect edge directions). Because all graphs are undirected, I use a modified version of the SHD that does not take into account incorrect edge directions. The True Positive Ratio measures how many of the true edges were correctly identified:

$$\text{TPR} = \frac{\text{True Positives}}{\text{Total True Edges in DAG}} \tag{7}$$

The False Positive Ratio measures how many incorrect edges were added:

$$\text{FPR} = \frac{\text{False Positives}}{\text{Total True Edges in DAG}} \tag{8}$$

The SHD metric captures overall structure error, while TPR and FPR measure how accurately each method reconstructs the true skeleton of the DAG. This allows a direct comparison of constraint-based and Bayesian methods in terms of their ability to recover the data-generating structure

2.3 Monte Carlo Design

I run 500 monte carlo iterations for each design setting (sample size N, network size n,network density d ,edge strength θ). Each of the 500 iterations starts by randomly generating a Bayesian Network (consistent with the design setting) and simulating data from this Gausian Bayesian Network. I then run the PC and the order MCMC algorithm on this data set and evaluate both the CPDAG and the undirected MAP DAG on all three evaluation metrics. In total I generate 500 realizations of each of the three evaluation metrics, for both of the two algorithms for each of the 48 design settings. This allows me to do a statistically robust comparison of the two algorithms across many different networks and sample size settings. [1]

[1] For the order-MCMC approach I use the Normal-wishart prior which results in the BGe Score for my multivariate normally distributed data. I run the default number of iterations in the mcmc chain which is $6n^2log(n)$. As a thinning procedure to control the autocorrelation, I also use the default setting which results in a total number of 1000 DAG's in the final posterior sample. For the conditional independence tests within the PC algorithm I use a standard significance level of $\alpha = 0.05$. Replication files are available here : GitHubRepo

3 Results

3.1 MCMC diagnostics

Before analyzing the Monte Carlo results I provide log likelihood trace plots to show the convergence behavior of the order-mcmc algorithm. Figure 1 plots 5 independent MCMC runs on the same

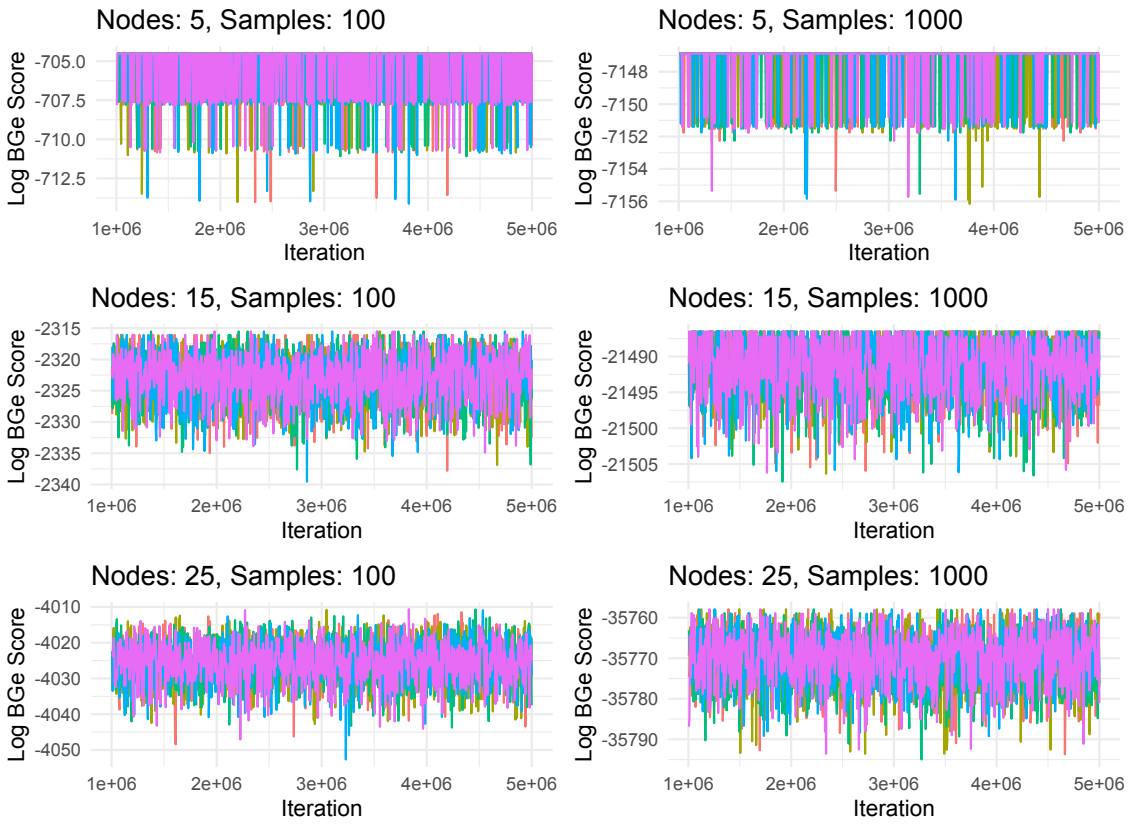

Figure 1: Likelihood Trace Plots of the order-MCMC algorithm across different sample designs

data set across different sample settings varying across network size and sample size. The burn-in period of the fist 20% of observations is excluded.
A well-mixed trace plot should show a smooth path without sudden jumps or long stretches where the BGe score stays the same. In the case of small networks with 5 nodes, the discrete nature of the DAG space is more pronounced, leading to sharp up-and-down movements in the log-score and poor mixing, making it difficult to assess convergence. As the network size increases, the trace plots appear smoother, indicating that the Markov chain explores the space more effectively and mixes well, suggesting better convergence behavior.

3.2 Structure Learning Performance

I begin by evaluating the performance of recovering undirected network structure measured by the SHD metric. Figure 2 compares the Structural Hamming Distance (SHD) across different simulation settings, with lower values indicating better performance. Each of the four plots in the figure isolates a single design parameter—sample size, network size, sparsity, or edge strength—while averaging over all other design settings. For example, the SHD value for a sample size of 10 in the upper left plot represents the mean SHD across all 500 Monte Carlo runs, aggregated over all tested network sizes, sparsity levels, and edge strengths. The central points in the plots indicate the mean SHD, while the vertical bars capture the variability in SHD by displaying the 10th and 90th percentiles of the distribution.

Figure 3 plots the false positive ratio (FPR) against the true positive ratio (TPR) in a scatterplot for different network size/sample size combinations. Within each network size/sample size combination, network density is varied, visualizing the differences in performance from learning sparse and dense network structures.

While Figure 2 shows the general trend of better performance with increased sample size for

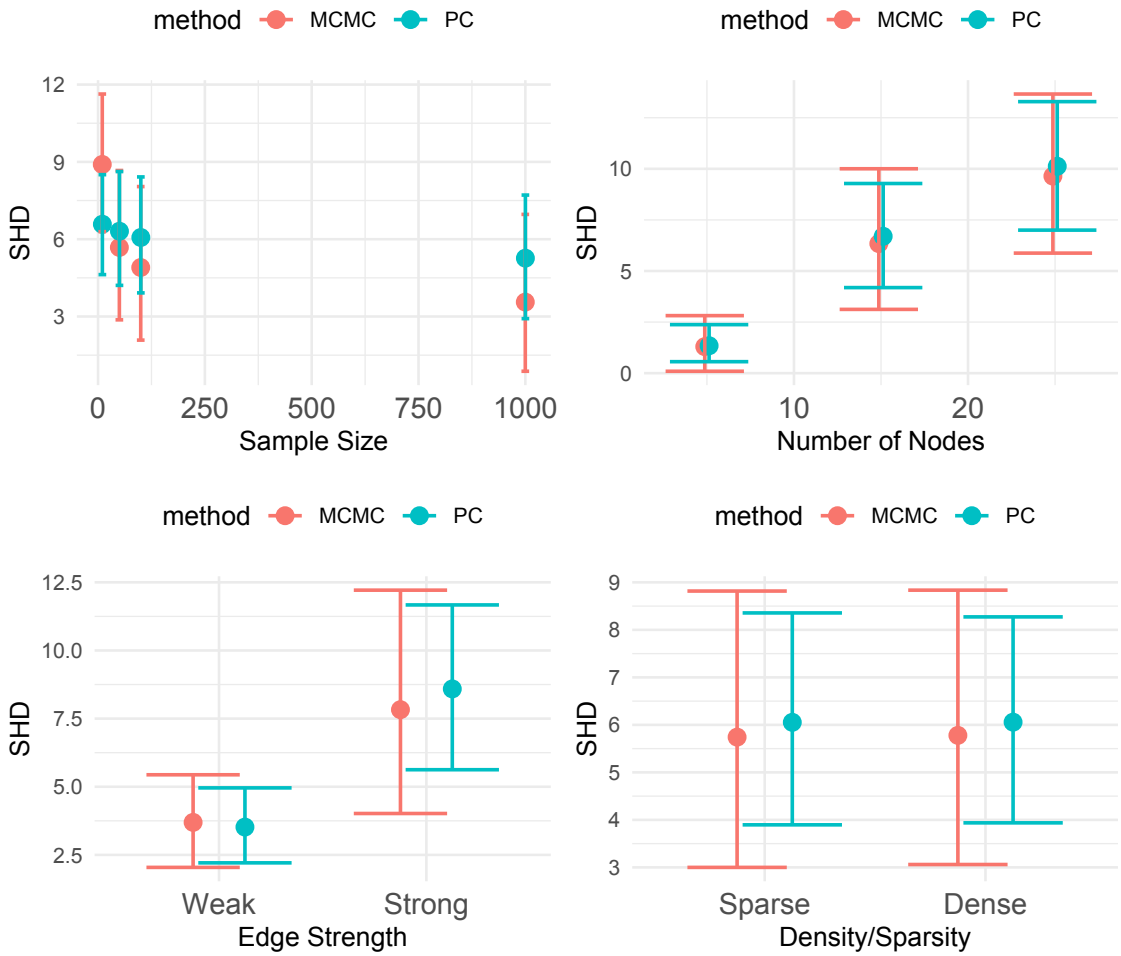

Figure 2: PC vs order-MCMC Performance on the Structural Hamming Distance (SHD) across different sample settings

both algorithms, it is surprising to see that both algorithms perform worse when the edge strength becomes stronger. The network density seems to have no performance effect at all. Going into the disaggregated data plotted in Figure 3, one can see that while in dense settings the algorithms generally find fewer true positives, they tend to find more false positives in more sparse settings. These two effects seem to cancel each other out in the aggregated effect displayed in the lower right plot in Figure 2. Figure 2 also shows that over all settings the PC algorithm has less variation than the order-MCMC.

Comparing the two algorithms, one sees that they perform very similarly overall and across different design settings. As previously discussed, the evaluation metrics plotted in Figures 2 and 3 measure performance only in recovering the undirected network structure of the CPDAG from the PC algorithm and the MAP DAG from the order-MCMC algorithm. These metrics do not account for the advantages of the Bayesian approach, such as full uncertainty quantification and the identification of fully directed DAGs. Therefore, the results should be interpreted with caution and do not necessarily indicate which approach is better for practical causal discovery.

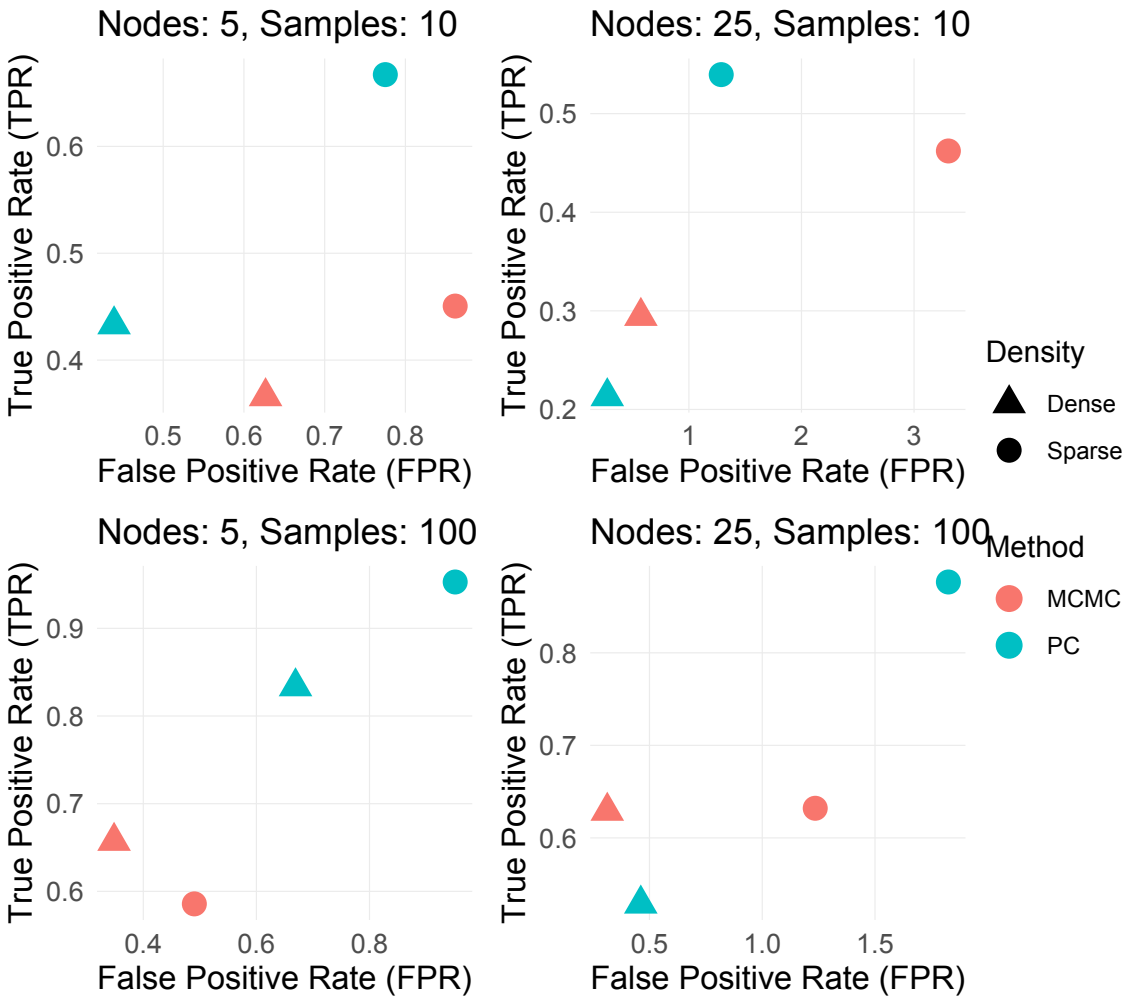

Figure 3: PC vs order-MCMC Performance regarding True & False Positive edges (TPR & FPR) across different sample settings

References

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

[Geiger and Heckerman, 2002] Geiger, D. and Heckerman, D. (2002). Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *The Annals of Statistics*, 30(5):1412–1440.

[Kalisch and Bühlman, 2007] Kalisch, M. and Bühlman, P. (2007). Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3).

[Kuipers et al., 2022] Kuipers, J., Suter, P., and Moffa, G. (2022). Efficient sampling and structure learning of bayesian networks. *Journal of Computational and Graphical Statistics*, 31(3):639–650.

[Madigan et al., 1995] Madigan, D., York, J., and Allard, D. (1995). Bayesian graphical models for discrete data. *International statistical review/revue internationale de statistique*, pages 215–232.

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

[Spirtes et al., 2001] Spirtes, P., Glymour, C., and Scheines, R. (2001). *Causation, prediction, and search*. MIT press.