Efficient Markov Network Discovery Using Particle Filters

Dimitris Margaritis
Dept. of Computer Science
Iowa State University
Ames, IA 50010

`dmarg@cs.iastate.edu`

Facundo Bromberg
Departamento de Sistemas de Información
Universidad Tecnológica Nacional
Facultad Regional Mendoza
Ciudad Mendoza, M5502AJE, Argentina

`fbromberg@frm.utn.edu.ar`

Abstract

In this paper we introduce an efficient independence-based algorithm for the induction of the Markov network structure of a domain from the outcomes of independence test conducted on data. Our algorithm utilizes a particle filter (sequential Monte Carlo) method to maintain a population of Markov network structures that represent the posterior probability distribution over structures, given the outcomes of the tests performed. This enables us to select, at each step, the maximally informative test to conduct next from a pool of candidates according to information gain, which minimizes the cost of the statistical tests conducted on data. This makes our approach useful in domains where independence tests are expensive, such as cases of very large data sets and/or distributed data. In addition, our method maintains multiple candidate structures weighed by posterior probability, which allows flexibility in the presence of potential errors in the test outcomes.

Keywords: Markov networks, particle filters, graphical model structure learning.

1   Introduction

In this paper we address the problem of learning the structure of **Markov networks** (**MNs**) from data. Markov networks are graphical statistical models whose structure can be represented by an undirected graph. (Other graphical models include Bayesian networks, represented by directed acyclic graphs.) A Markov network consists of two parts: an undirected graph (the model structure), and a set of parameters. An example Markov network is shown in Figure 1, for a domain containing eight variables (also called attributes). Learning such models from data consists of two interdependent problems: learning the structure of the network, and, given the learned structure, learning the parameters. In this work we focus on structure learning of the MN from data, which is frequently the most challenging of the two tasks.

The structure of a Markov network graphically encodes a set of conditional independences among the variables in the domain. Knowledge of these independences is invaluable in a number of fields, especially those that rely more on qualitative than quantitative models (e.g., social sciences). Markov networks have also been used in the physics and computer vision communities (Geman and Geman, 1984; Besag et al., 1991; Anguelov et al., 2005) where they have been historically called Markov random fields. Recently there has been interest in their use for spatial data mining, which has applications in geography, agriculture, climatology, ecology and others (Shekhar et al., 2004).

There exist two broad classes of algorithms for learning the structure of graphical models: *score-based* (Heckerman, 1995) and *independence-based* or *constraint-based* (Spirtes et al., 2000).
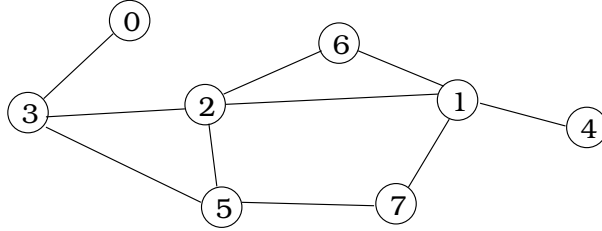
Figure 1: Example Markov network. The nodes represent variables in the domain $\mathbf{V} = \{0, 1, 2, 3, 4, 5, 6, 7\}$.

Score-based approaches conduct a search in the space of legal structures (of size super-exponential in the number of variables in the domain) in an attempt to discover a model structure of maximum score. As such, heuristic search methods are frequently necessary. Independence-based algorithms instead exploit the fact that a graphical model structure implies that a set of independences must exist in the distribution of the domain, and therefore in the data set provided as input to the algorithm (under the certain assumptions, see next section); they work by conducting a set of statistical conditional independence tests on data, successively restricting the number of possible structures consistent with the results of those tests to a singleton (if possible), and inferring that structure as the only possible one.

In this work we present an algorithm that belongs to the latter category. A central objective of our algorithm is to minimize the total cost of conditional tests evaluated on data; this represents advantages in domains where independence tests are expensive. One case where this may occur is applications involving very large data sets (number of data points). Another case is domains where the data are heterogeneously distributed i.e., where columns of the data set (attributes) may be located in geographically distinct locations (e.g., sensor networks, weather modeling and prediction, traffic monitoring, etc). In such settings, conducting a conditional independence test is very expensive, involving transfers of large amounts of data over a possibly slow network, so it is important to minimize the cost of tests done.

The rest of the paper is organized as follows: In the next section we present our notation, followed by related work in Section 3. In Section 4 we present our approach in detail, and in Section 5 we evaluate our approach, comparing it to other independence-based approaches from the literature. We conclude and summarize our contributions in Section 6.

2   Notation and Assumptions

A Markov network of a domain is an undirected model that can be used (i) to represent the joint probability distribution of the domain, and (ii) to represent the set of conditional independences present in the domain. A Markov network consists of a structure and a set of parameters. The parameters are clique potentials (Pearl, 1988). The application domain $\mathbf{V}$ is a set of random variables of size $n = |\mathbf{V}|$. In this work we use capital letters $A, B, \ldots$ to denote domain random variables, small letters to denote their values (e.g., $a, b, \ldots$) and bold letters for sets of variables (e.g., $\mathbf{S}$). The space of all structures (given $\mathbf{V}$) is denoted by $\mathcal{X}$ and the space of all conditional independence tests by $\mathcal{Y}$. Each variable in $\mathbf{V}$ is represented by a node in the Markov network of the domain. Conditional independence of variables A and B given the set of variables $\mathbf{S}$ is denoted by $(A \perp\!\!\!\perp B \mid \mathbf{S})$, while conditional dependence by $(A \not\!\perp\!\!\!\perp B \mid \mathbf{S})$. The structure of a Markov network implies that a number of conditional independences are present in the underlying probability distri-

bution of the domain; these independences are at least those implied by vertex separation through the network structure i.e., $(A \perp\!\!\!\perp B \mid \mathbf{S})$ if A and B are separated in the MN graph after removing each node corresponding to a variable in $\mathbf{S}$ (including all edges incident on them). For example, in Figure 1, $(0 \perp\!\!\!\perp 4 \mid \{2, 7\})$. Depending on the values of the parameters of the network additional independences may exist, even between variables connected by an edge. If the set of conditional independences implied by a graph G (using the vertex separation procedure just described) are *exactly* those present in some probability distribution P then we say that P and G are *faithful* to one another (Spirtes et al., 2000), and that P satisfies the Faithfulness property. The property of Faithfulness excludes certain distributions that are unlikely to happen in practice, and is needed for proofs of correctness. It is therefore a common assumption of independence-based algorithms for graphical model discovery. We assume Faithfulness in the present work.

As described later in our main algorithm, we maintain populations of structures, called particles, at each time step t; slightly abusing our notation we denote these particle populations by $\mathcal{X}_t$. We also denote a sequence of t tests $Y_1, \ldots, Y_t$ by $Y_{1:t}$ and a sequence of value assignments to these tests (independence or dependence, corresponding to `true` and `false` respectively) by $y_{1:t}$.

3 Related Work

As mentioned above, one class of algorithms for learning the structure of graphical models is the *score-based* approach, exemplified for Markov networks by Della Pietra et al. (1997); McCallum (2003). Score-based approaches conduct a search in the space of legal structures in an attempt to discover a model structure of maximum score. This score is usually a function of the data likelihood, or more often a penalized version of it e.g., minimum description length (Lam and Bacchus, 1994), or pseudo-likelihood (Besag, 1974). Due to the intractable size of the search space i.e., the space of all legal graphs, which is super-exponential in size, score-based algorithms must usually resort to heuristic search. At each step of the structure search, a probabilistic inference step is necessary to evaluate the score. For Bayesian networks this inference step is tractable and therefore several practical score-based algorithms for structure learning have been developed (Lam and Bacchus, 1994; Heckerman, 1995). For undirected models however, probabilistic inference requires the calculation of a normalizing constant (also known as the partition function), a problem known to be NP-hard (Jerrum and Sinclair, 1993; Barahona, 1982). A number of approaches have considered a restricted class of graphical models e.g., Chow and Liu (1968); Rebane and Pearl (1989); Srebro and Karger (2001). However, Srebro and Karger (2001) proves that finding the maximum likelihood network is NP-hard for Markov networks of tree-width greater than 1.

Some work in the area of structure learning of undirected graphical models has concentrated on the learning of decomposable (also called chordal) MNs (Srebro and Karger, 2001). An example of learning non-decomposable MNs is presented by Hofmann and Tresp (1998), which is an approach for learning structure in continuous domains with non-linear relationships among the domain attributes. Their algorithm removes edges greedily based on a leave-one-out cross validation log-likelihood score. A non-score-based approach is Abbeel et al. (2006), which introduces a new class of efficient algorithms for structure and parameter learning of factor graphs, a class of graphical models that subsumes Markov and Bayesian networks. Their approach is based on a new parameterization of the Gibbs distribution in which the potential functions are forced to be probability distributions, and is supported by a generalization of the Hammersley-Clifford theorem for factor graphs. It is a promising and theoretically sound approach that may lead in the future to practical and efficient algorithms for undirected structure learning. More recent score-based work includes a method for learning Markov network structure using an L_1 regularization norm over the

parameter space (Lee et al., 2007), which has been shown to have both theoretical advantages over the usual L_2 norm in terms of sample complexity and good generalization performance in practice. This work presents a departure of the standard approach of a heuristic search in the space of structures, using instead a method that converts the problem into a convex optimization problem over the joint distribution of data and parameters in a log-linear model, where missing edges correspond to cases when certain parameters are zero.

In this work we present algorithms that belong to the *independence-based* or *constraint-based* approach (Spirtes et al., 2000). Independence-based algorithms exploit the fact that a graphical model implies that a set of independences exist in the distribution of the domain, and therefore in the data set provided as input to the algorithm (under the Faithfulness assumption, see previous section); they work by conducting a set of conditional independence tests on data, successively restricting the number of possible structures consistent with the results of those tests to a singleton (if possible), and inferring that structure as the only possible one. A desirable characteristic of independence-based approaches is the fact that they do not require the use of probabilistic inference during the process of discovery of the structure. Also, such algorithms are amenable to proofs of correctness (under assumptions). In general, score-based approaches may be better suited for prediction or inference tasks due to their optimization of a function of the likelihood while independence-based ones for tasks where understanding the interactions among variables in a domain is more important, e.g., econometrics, psychology, or sociology. Whether this is true for various algorithms in the corresponding classes may depend on various factors (e.g., the penalty factor in score-based approaches, corresponding to a prior on the parameters, or the order that independence tests are executed in independence-based methods). More importantly it may depend on the nature of the application for which the resulting structure is to be used i.e., whether the Markov network is viewed purely as a predictive tool or whether insights into the structure of the underlying generative model have the greatest importance.

For Bayesian networks, the independence-based approach has been mainly exemplified by the SGS (Spirtes et al., 2000), PC (Spirtes et al., 2000), and algorithms that learn the Markov blanket as a step in learning the Bayesian network structure such as Grow-Shrink (GS) algorithm (Margaritis and Thrun, 2000), IAMB and its variants (Tsamardinos et al., 2003a), HITON-PC and HITON-MB (Aliferis et al., 2003), MMPC and MMMB (Tsamardinos et al., 2003b), and max-min hill climbing (MMHC) (Tsamardinos et al., 2006), all of which are widely used in the field. Algorithms for restricted classes such as trees (Chow and Liu, 1968) and polytrees (Rebane and Pearl, 1989) also exist.

Previous work on learning Markov networks has mainly focused on learning *Gaussian graphical models*, where the assumption of a continuous multivariate Gaussian distribution is made; this results in linear dependences among the variables with Gaussian noise (Whittaker, 1990; Edwards, 2000). This has been relaxed by the GSMN and GSIMN algorithms of Bromberg, Margaritis, and Honavar (2006), which apply to any case where an arbitrary faithful distribution can be assumed and a probabilistic conditional independence test for that distribution is available. For instance, they can be applied for the case of continuous Gaussian variables with the use of partial correlation as independence test. Similar to some Bayesian networks learning algorithms mentioned above (e.g., IAMB, HITON-PC, etc.), the GSMN algorithm was inspired by and works in a fashion similar to the GS algorithm of Margaritis and Thrun (2000). The GSIMN algorithm augments GSMN by the use of Pearl's inference axioms Pearl (1988) to infer the result of certain independence tests without actually performing them. Compared to the approach described in this work, both GSMN and GSIMN have two limitations: (i) potential inefficiencies with regard to the number of tests required to learn the structure due to the relatively rigid (predefined) order in which tests are performed, and (ii) potential instability due to cascading effects of errors in the test results. Instead, the present

paper takes a Bayesian approach that maintains the posterior probability distribution over the space of structures given the tests performed so far. We avoid the inefficiencies of previous approaches by greedily selecting, at each step, the optimally informative tests according to information gain (decrease in entropy of the posterior distribution). As such the approach can be seen as an instance of active learning (Tong and Koller, 2001). We compare our approach to both GSMN and GSIMN algorithms in Section 5.

Our algorithm is an implementation of a particle filter or sequential Monte Carlo (SMC) approach (Doucet et al., 2001), with some important differences from their typical application. In particular, while SMC methods usually model an evolving distribution over the state of some important variable (e.g., the location of a target in a radar tracking application), the distribution over the variable of interest in our case (the Markov network structure) is static i.e., all observations are assumed to have been produced from the same distribution (a Dirac function concentrated on the "correct" structure) even though we consider a sequence of distributions after taking into account one observation at a time (gathered from a statistical test). This has many similarities with existing work in the SMC literature. For example, Chopin (2002) purposefully creates a sequence of artificial distributions, each conditional on an increasing number of observations, the last of which is the distribution of interest. A number of approaches addressing the problem of probability estimation and inference over a common space include Del Moral et al. (2006); Cappé et al. (2004).

In the following section we present our approach in detail.

4   Approach

As mentioned above, the structure of a faithful Markov network encodes a value for every possible conditional independence statement that can be formed by variables of the domain. This property points to one possible way of learning the structure of the Markov network of the domain, namely examining the values of (a subset of) these independence statements; this is the essence of the independence-based approach to structure learning. Certain challenges exist in such approaches. One is the fact that each test, when conducted on a finite data set, can only return a partial evidence of the existence of the corresponding independence (or dependence). In other words, given a data set D, one may be able to calculate the posterior probability of independence or dependence of test Y i.e., $\Pr(Y \mid D)$, whose value is typically not exactly 0 or exactly 1. Given such evidence, most independence-based structure algorithms proceed by assuming the most probable value as the correct one (i.e., *thresholding* the value), and eliminate all structures that are inconsistent with this value. However, this may discard important information, as it ignores the *degree of agreement* of each structure with the tests conducted at each point in the execution of the algorithm. One way to represent this degree of agreement is by the posterior probability over the space of all structures, given the tests conducted so far. Doing so allows one to select, at each point in the execution, the most probable structure, which corresponds to the one that agrees with these test results the most. It also makes it possible to retain structures that disagree with the most probable value of some independence test, as one of these may in fact end up being the one of that agrees with the *entire* set of tests the most i.e., it may be *collectively* the most probable structure given all tests done (even if it disagrees with the most probable value of some of the tests). Another problem of thresholding is the possibility of errors in the tests when conducted on a data set. This is a serious problem for algorithms that use this methodology, as even a single error may eliminate the correct structure. By maintaining instead the posterior probability distribution, one preserves the possibility that the correct structure is chosen at the end (if it ends up being the most probable one). A final benefit is the fact that maintaining a distribution over structures makes it possible
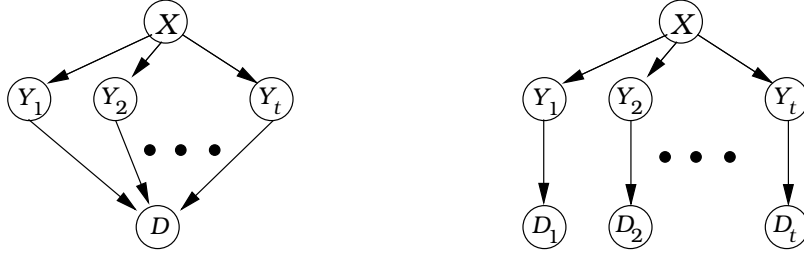
Figure 2: Generative model of domain. **Left:** Ideal. **Right:** Assumed.

to select tests to conduct in a way that maximizes the information that can be expected to be obtained by them.

For all these reasons, in this paper we take such an approach, called the Bayesian approach, which maintains the posterior probability over structures given the values of tests conducted so far. To be able to calculate this probability in a principled manner, we must first define a generative model that precisely represents the interactions between the quantities involved i.e., the structure, the test outcomes, and the data; this is what we present in the next section. We also show that the ideal model is in practice intractable, and present an approximation that allows the analytical calculation of the posterior distribution.

4.1 Generative Model over Structures and Tests

Given an input data set D, $\Pr(X \mid D)$ is the posterior probability over structures that takes into account the data in D. The variable X models the structure and its domain is $\mathcal{X}$, the set of all possible structures. Independence-based algorithms extract information from the data set in the form of statistical independence tests; we indicate the tests conducted from time step 1 through t by $Y_1, Y_2, \ldots, Y_t$, abbreviated $Y_{1:t}$. Each variable Y_i represents a single conditional independence test and its domain is binary, with $Y_i = \texttt{true}$ ($Y_i = \texttt{false}$) corresponding to the state of the world in which test Y_i evaluates to independence (dependence). Variable D models the data set, whose domain is the set of all possible data sets.

A central assumption made by independence-based algorithms is that the tests conducted are the *sufficient statistics* for the structure i.e., there is no information in the data set beyond the value of the tests as far as model structure is concerned. This stems from the fact that the structure X faithfully encodes the independences in the domain. A generative model that encodes this assumption is shown in Figure 2 (left). (Note that this assumption is not particular to our approach, but is implicit in any independence-based approach. Our model only makes it explicit and formalizes it.) We call this the *ideal model*. Note that in the ideal model variables X and D are shown d-separated (rendered conditionally independent) by the tests $Y_1, \ldots, Y_t$ done at a given time t during the execution of our algorithm i.e., $(X \perp\!\!\!\perp D \mid Y_{1:t})$; this is precisely what encodes the fact that these tests are the sufficient statistics for the data set.

This ideal generative model presents a number of challenges; one problem is that the posterior over structures $\Pr(X \mid D)$ cannot be computed in practice; this is because, according to the model, $\Pr(X \mid D) = \sum_{y_{1:t}} \Pr(X \mid Y_{1:t} = y_{1:t}, D) \Pr(Y_{1:t} = y_{1:t} \mid D)$, which requires the computation of $\Pr(Y_{1:t} = y_{1:t} \mid D)$, the *joint* outcome of a set of tests, currently an unsolved problem (Miller, 1981; Benjamini and Hochberg, 1995; Benjamini and Yekutieli, 2001). In practice however we can only compute the probability of the outcome of a single test. To overcome this dependence among the test outcomes we therefore assume a simplified model, shown in Figure 2 (right). This model contains multiple data sets $D_1, \ldots, D_t$ (abbreviated $D_{1:t}$) which are independent given tests $Y_{1:t}$. This allows

6

the model to be solved analytically because now $\Pr(Y_{1:t} = y_{1:t} \mid D) \propto \Pr(D \mid Y_{1:t} = y_{1:t}) \Pr(Y_{1:t} = y_{1:t})$, where $D = \bigcup_{i=1}^{t} D_i$, and the first factor decomposes as the product $\prod_{i=1}^{t} \Pr(D_i \mid Y_i = y_i)$. Furthermore, the factors $\Pr(D_i \mid Y_i = y_i)$ can be computed by known procedures such as the discrete version of the Bayesian test of Margaritis (2005), which computes $\Pr(Y_i = y_i \mid D_i)$ by analytically calculating the data likelihood $\Pr(D_i \mid Y_i = y_i)$ of two competing multinomial models corresponding to $y_i = \texttt{true}$ and $y_i = \texttt{false}$ (the independent and the dependent model), i.e., $\Pr(D_i \mid Y_i = \texttt{true})$ and $\Pr(D_i \mid Y_i = \texttt{false})$. The calculation of these likelihoods is shown in Appendix B, while a detailed derivation of an expression for the posterior $\Pr(X \mid D)$ is presented in the next section. Note that in the assumed model, as in the ideal one, variables X and D are d-separated by the tests $Y_1, \dots, Y_t$, following the sufficient statistics principle mentioned above.

In practice we do not have more than one data set, and therefore we use the same data set for all tests i.e., $D_i = D_j$, for all i, j. Thus, the model depicted on Figure 2 (right) is used only as an approximation, for the purpose of overcoming the lack of an exact solution described above. As we will show in the experiments section, this approximation works reasonably well in both artificial and real-world data sets.

4.2 Exact Posterior Probability Calculation under Assumed Model

Before we begin, we introduce some additional helpful notation: we abbreviate the posterior probability over structures given data sets $d_{1:t}$ as $\Pr_t(X)$, the posterior probability over structures at time t. Also, in our calculations below we use the conditional entropy of X given the set of tests $Y_{1:t}$ and the data sets $d_{1:t}$, $H(X \mid Y_{1:t}, d_{1:t})$, similarly abbreviated as $H_t(X \mid Y_{1:t})$.

As mentioned above, to learn the MN structure of a domain from data, we employ a Bayesian approach, calculating the posterior probability $\Pr_t(x) = \Pr(X = x \mid d_{1:t})$ of a structure $x \in \mathcal{X}$ at time t given data sets $d_{1:t}$. We now derive an expression for this posterior.

By Bayes' law we have:
$$\Pr(x \mid d_{1:t}) \propto \Pr(x) \Pr(d_{1:t} \mid x)$$

which, by law of total probability for variables $Y_{1:t}$ equals

$$\Pr(x \mid d_{1:t}) \propto \Pr(x) \sum_{y_{1:t}} \Pr(d_{1:t} \mid y_{1:t}, x) \Pr(y_{1:t} \mid x). \tag{1}$$

According to our Faithfulness assumption, a structure x encodes exactly those independences that hold in the underlying probability distribution i.e., under this assumption the state of a set of conditional independence tests $Y_{1:t}$ is completely determined by x. In terms of probabilities, this is equivalent to saying that $\Pr(Y_{1:t} \mid X = x) \in \{0, 1\}$. That is, there exists only a single assignment to each test in $Y_{1:t}$ whose probability, conditioned on x, is non-zero. We denote this assignment by $y_{1:t}^x$, and we say that x is *consistent* with assignment $y_{1:t}$ when it equals $y_{1:t}^x$. These considerations can be condensed in the following expression:

$$\Pr(Y_{1:t} = y_{1:t} \mid X = x) = \delta(y_{1:t}, y_{1:t}^x), \tag{2}$$

where $\delta(a, b)$ is the Kronecker delta function that equals 1 if and only if a is equal to b, and 0 otherwise.

Note that even though a structure determines the assignment $y_{1:t}^x$ of a sequence of tests $Y_{1:t}$, the reverse does not hold in general i.e., given an assignment $y_{1:t}$, there may be more than one structure consistent with it. If two structures x and x' are consistent with the same assignment $y_{1:t}$ we say they are *equivalent with respect to test results* $y_{1:t}$ and we denote this fact by $x \sim_{y_{1:t}} x'$.

This generates an equivalence relation that partitions the space $\mathcal{X}$ into *equivalence classes*, one for each possible assignment $y_{1:t}$, which we denote by $\{y_{1:t}\}$ and define it formally by

$$\{y_{1:t}\} = \{x \in \mathcal{X} \mid x \text{ is consistent with } y_{1:t}\}. \tag{3}$$

In what follows, we will denote the class of structures equivalent to structure x with respect to tests $Y_{1:t}$ (which have values $y_{1:t}^x$ in x) by $\{y_{1:t}^x\}$.

Returning to the derivation of the posterior distribution, we apply Eq. (2) to Eq. (1) to obtain $\Pr(y_{1:t} \mid x) = \delta(y_{1:t}, y_{1:t}^x)$ and thus only the term $y_{1:t}^x$ survives in the sum i.e.,

$$\Pr(x \mid d_{1:t}) \propto \Pr(x) \Pr(d_{1:t} \mid y_{1:t}^x, x).$$

Finally, using the generative model of the previous section (Figure 2 (right)), we obtain

$$\Pr(x \mid d_{1:t}) \propto \Pr(x) \prod_{i=1}^{t} \Pr(d_i \mid y_i^x). \tag{4}$$

The constant of proportionality in the above equation is independent of x and thus can be calculated by a sum over all structures in $\mathcal{X}$. Since this space is super-exponential in size, this requires an approximation. In the next section we present a principled procedure to approximate this and other quantities that require a summation over $\mathcal{X}$.

The prior $\Pr(x)$ can in principle be arbitrary. In our experiments, presented later in Section 5, we chose it in a way that provides equal representation to dense as well as sparse networks. In particular, one can partition the space $\mathcal{X}$ of all structures in $\binom{n}{2}$ subspaces $\mathcal{X}_h, h = 1, \ldots, \binom{n}{2}$, where $\mathcal{X}_h = \left\{ x = (\mathbf{V}, E) \in \mathcal{X} \mid |E| = h \right\}$ consists of all structures with exactly h edges. The aforementioned prior $\Pr(X)$ is uniform over the set of subspaces $\{\mathcal{X}_h\}$ of $\mathcal{X}$. Therefore, the prior probability of structure $x = (\mathbf{V}, E)$ is

$$\Pr(X = x) = \frac{1}{\binom{n}{2}} \frac{1}{|\mathcal{X}_{|E|}|}. \tag{5}$$

As the size $|\mathcal{X}_h|$ of a subspace $\mathcal{X}_h$ is equal to $\binom{\binom{n}{2}}{h}$, some subspaces may be exponentially larger than others e.g., $\mathcal{X}_{\binom{n}{2}/2}$ vs. $\mathcal{X}_1$. Thus, this choice for the prior is made to facilitate the sufficient exploration of small portions of the space such as $\mathcal{X}_1$ as well as large ones like $\mathcal{X}_{\binom{n}{2}/2}$.

Finally, the quantity $\Pr(d_i \mid y_i^x)$ in Eq. (4) is the likelihood of the data given the assignment y_i^x to test Y_i. To compute these quantities we use the Bayesian test described in Margaritis (2005), adapted for the discrete case. This test analytically calculates and compares the likelihoods of two competing multinomial models (with different numbers of parameters), namely $\Pr(d_t \mid Y_t = y_t)$, for $y_t \in \{\texttt{true}, \texttt{false}\}$. A derivation of these likelihoods using the aforementioned test is contained in Appendix B. We are not aware for any other work in the literature for these likelihoods.

We now give a general overview of our approach for representing the approximation of the posterior probability over structures. This is followed by a detailed description of our algorithm in Section 4.4.

4.3 Approximation of the Posterior over Structures via a Particle Filter

According to the independence-based approach, the problem of learning the structure of a Markov network consists on finding a sequence of $t^\star$ tests $Y_{1:t^\star}^\star$ of minimum cost (the cost of each test is

described in Section 4.4.2 later in the paper), such that only a single structure $x^\star$ is consistent with the outcome $y_{1:t^\star}$ of these tests. This is always theoretically possible due to our assumption of Faithfulness, which guarantees the existence of a single structure consistent with the results of all possible tests in the domain. The situation in which a single structure $x^\star$ is consistent with the tests can be represented in our framework with a posterior that concentrates all probability mass in $x^\star$, i.e., $\Pr_{t^\star}(X = x^\star) = 1$ and $\Pr_{t^\star}(X \neq x^\star) = 0$. This posterior distribution has conditional entropy 0, i.e. $H(X \mid Y^\star_{1:t^\star}, d_{1:t^\star}) = 0$. We can therefore summarize the problem of learning a structure in this framework by the following two steps:

1. Finding a sequence of tests $Y^\star_{1:t^\star}$ of minimum cost such that $H(X \mid Y^\star_{1:t^\star}, d_{1:t^\star}) = 0$, and

2. Finding the (unique) structure $x^\star$ such that $\Pr(X = x^\star \mid d_{1:t^\star}) = 1$.

In practice, the above procedure presents considerable difficulties because:

- The space of structures $\mathcal{X}$ is super-exponential: $|\mathcal{X}| = 2^{\binom{n}{2}}$. Thus, the exact computation of the entropy $H_t(X \mid Y_{1:t})$, a sum over all $x \in \mathcal{X}$, is intractable.

- The space of candidate tests $\mathcal{Y}$ is also at least exponential in size: there are $\binom{n}{2}\binom{n-2}{m}$ possible tests $(A \perp\!\!\!\perp B \mid \mathbf{S})$ with $|\mathbf{S}| = m$, and m ranges from 0 to $n - 2$. Moreover, for a given number of tests t, there exist $\binom{|\mathcal{Y}|}{t}$ possible candidate test outcome sequences $y_{1:t}$ to sum over in the entropy calculation.

We address the first issue using a **particle filter**, also called a **sequential Monte Carlo (SMC)** approach (Doucet et al., 2001). At each step t, we maintain a population $\mathcal{X}_t$ of candidate MN structures (called *particles*) for the purpose of representing the posterior probability distribution over structures given the outcomes of tests performed so far. In this way, all required quantities, such as posterior probability $\Pr_t(x)$ or conditional entropy $H_t(X \mid Y_{1:t})$, can be estimated by simple averaging over the particle population $\mathcal{X}_t$. The theory of Monte Carlo sampling provides a principled procedure for approximating these summations using a sample $\{x^{(i)}\}_{i=1}^N$ for the space $\mathcal{X}$. Formally, the Monte Carlo principle (Andrieu et al., 2003, p. 5) states that, given an i.i.d. sample $\{x^{(i)}\}_{i=1}^N$ from probability distribution $p(x)$, it is the case that,

$$I_N(f) = \frac{1}{N} \sum_{i=1}^N f(x^{(i)}) \xrightarrow[N \longrightarrow \infty]{a.s.} I(f) = \sum_{x \in \mathcal{X}} f(x) p(x) \tag{6}$$

for some measurable function $f(x)$. The estimate $I_N(f)$ is unbiased and, by the strong law of large numbers converges to $I(f)$ almost surely (*a.s.*). To illustrate, let us approximate the normalization constant k of Eq. (4) (i.e., $\Pr(x \mid d_{1:t}) = \frac{\Pr(x)\Pr(d_{1:t}|x)}{k}$) which involves a summation over all structures in $\mathcal{X}$, using the set of particles $\mathcal{X}_t = \{x^{(i)}\}_{i=1}^N$ at time t:

$$k = \sum_{x \in \mathcal{X}} \Pr(x) \Pr(d_{1:t} \mid x) \approx \frac{1}{N} \sum_{i=1}^N \Pr(d_{1:t} \mid x^{(i)}).$$

We address the second issue mentioned above (choosing the next test to perform) by searching for a candidate test using a greedy search procedure. At each step $t+1$ of our algorithm, we choose as the next test to perform the member $Y^\star_{t+1}$ of $\mathcal{Y}$ that minimizes the expected entropy $H_t(X \mid Y^\star_{1:t}, Y^\star_{t+1})$ (or, equivalently, maximizes the information gain), penalized by a factor proportional to its cost. Since $\mathcal{Y}$ is exponential in size, the minimization is performed through a heuristic search approach. The next section describes our main algorithm, called PFMN, and these steps in detail.

9

4.4 The PFMN Algorithm

Our algorithm is called **PFMN** (**Particle Filter Markov Network structure learner**), and is shown in Algorithm 1. At each time step t, the algorithm maintains a set $\mathcal{X}_t$ containing N structure particles.

Initially, each structure in $\mathcal{X}_0$ is generated by sampling from the prior distribution $\Pr(x)$ of Eq. (5) (line 1). This is done in two steps. First the cardinality m is selected uniformly from $[1, \binom{n}{2}]$. Second, a structure x with cardinality m is selected uniformly from the set of all structures with cardinality m. This is done by selecting as edges of x the first m pairs from a random permutation of all pairs (i, j), where $i, j \in \{1, \cdots, n\}$ and $i \neq j$.

Algorithm 1 Particle Filter Markov Network (PFMN) algorithm. $x = PFMN(N, M, q(X^\star \mid X))$

1: $\mathcal{X}_0 \longleftarrow$ sample N independent particles distributed according to prior $\Pr(x)$ (c.f. Eq.(5)).
2: $t \longleftarrow 0$
3: **while** forever **do**
4:    $Y_{t+1}^\star \longleftarrow OptimalTest(\mathbf{V}, score)$ /* Compute $\arg\max_{(A, B \mid \mathbf{S})} score(A, B \mid \mathbf{S})$ using Algorithm 2. */
5:    $Y_{1:t+1} \longleftarrow Y_{1:t} \cup \{Y_{t+1}^\star\}$
6:    $p_{\mathtt{t}} \longleftarrow \Pr(d_{t+1} \mid Y_{t+1}^\star = \mathtt{true})$ /* Data likelihood from statistical test on data. */
7:    $p_{\mathtt{f}} \longleftarrow \Pr(d_{t+1} \mid Y_{t+1}^\star = \mathtt{false})$ /* Data likelihood from statistical test on data. */
8:    Update $\Pr_{t+1}(X)$ from $p_{\mathtt{t}}$ and $p_{\mathtt{f}}$ using Eq. (4).
9:    $\mathcal{X}_{t+1} \longleftarrow PF(\mathcal{X}_t, M, \Pr_{t+1}(X), q(X^\star \mid X))$
10:   **if** $H_{t+1}(X \mid Y_{1:t+1}) \leq \epsilon$ **then**
11:      **return** $\arg\max_{x \in \mathcal{X}_t} \Pr_t(x)$ /* Return most probable structure. */
12:   $t \longleftarrow t + 1$

At each time t during the main loop of the algorithm (lines 3–12), the test $Y_{t+1}^\star = (A^\star, B^\star \mid \mathbf{S}^\star) \in \mathcal{Y}$ that optimizes a *score function* is selected. Since for each pair of variables $(A, B) \in \mathbf{V} \times \mathbf{V}$ the space of possible conditioning sets is exponential (equaling the power set of $\mathbf{V} - \{A, B\}$), this optimization is performed by heuristic search. The score function and the optimization procedure are described in detail in the next two sections.

The main loop of the PFMN algorithm continues by computing the data likelihoods of the optimal test $Y_{t+1}^\star$ in lines 6 and 7, which are used to update the posterior over structures and obtain the new posterior $\Pr_{t+1}(X)$ using Eq. (4). To represent this updated distribution, a new set of particles $\mathcal{X}_{t+1}$ is generated in line 9, using the particle filter algorithm described later in section 4.4.3.

The PFMN algorithm terminates when the entropy $H_t(X \mid Y_{1:t})$ (estimated over the population $\mathcal{X}_t$) is smaller than some small threshold $\epsilon \geq 0$. (In our experiments we used $\epsilon = 0$.) This occurs when each equivalence class over $Y_{1:t}$ contains few distinct structures i.e., most particles in it coincide, with the case of $\epsilon = 0$ corresponding to the limiting case where all particles in each equivalence class represent one single structure. When the termination condition is satisfied, the algorithm returns the most probable structure.

4.4.1 Optimal Test Selection Procedure

We now describe the algorithm for selecting the next test to perform, used in line 4 of the PFMN algorithm. The algorithm is shown in Algorithm 2, which takes as input the set of variables in the domain $\mathbf{V}$ and a score function *score*. Since the space of tests $\mathcal{Y}$ is super-exponential in size, i.e., for each pair of variables $(A, B) \in \mathbf{V} \times \mathbf{V}$ the space of possible conditioning sets is exponential (equal to $\mathbf{2}^{\mathbf{V} - \{A, B\}}$, the power set of $\mathbf{V} - \{A, B\}$), this optimization is performed by heuristic search, which selects the test $Y_{t+1}^\star = (A^\star, B^\star \mid \mathbf{S}^\star) \in \mathcal{Y}$ that optimizes (locally) the score function.

Algorithm 2 Computes test with optimal score. $(A^\star, B^\star \mid \mathbf{S}^\star) = OptimalTest(\mathbf{V}, score)$

1: $(A^\star, B^\star \mid \mathbf{S}^\star) \leftarrow \texttt{nil}.$
2: /* Find overall optimal test by performing a search over all pairs of variables.   */
3: **for all** $A, B \in \mathbf{V}, A \neq B$ **do**
4:     /* Search for conditioning set $\mathbf{S}$ that maximizes $score(A, B \mid \mathbf{S})$.   */
5:     $k \leftarrow 0$
6:     **repeat**
7:       $k \leftarrow k + 1$
8:       $\mathbf{S} \leftarrow \varnothing$
9:       **repeat** /* Find local maximum for sets k "flips" away. */
10:         $\Sigma_k \leftarrow \{\mathbf{S}' \in \mathbf{2}^{\mathbf{V}-\{A,B\}} \mid hamming(\mathbf{S}', \mathbf{S}) = k\}$ /* Σ_k is set of neighbors of $\mathbf{S}$. */
11:         $\mathbf{S}_{old} \leftarrow \mathbf{S}$
12:         **for all** $\mathbf{S}' \in \Sigma_k$ **do** /* Find optimal neighbor. */
13:           **if** $score(A, B \mid \mathbf{S}') > score(A, B \mid \mathbf{S})$ **then** $\mathbf{S} \leftarrow \mathbf{S}'$
14:       **until** $(\mathbf{S} = \mathbf{S}_{old})$ /* If set $\mathbf{S}$ has not changed, it is a local maximum: stop. */
15:     **until** $(k > K$ or $\mathbf{S} \neq \varnothing)$ /* Continue while the initial set $(\varnothing)$ is a local maximum. */
16:     /* Record current test $(A, B \mid \mathbf{S})$ if its better than optimal. */
17:     **if** $(A^\star, B^\star \mid \mathbf{S}^\star) = \texttt{nil}$ **then**
18:       $(A^\star, B^\star \mid \mathbf{S}^\star) \leftarrow (A, B \mid \mathbf{S})$
19:     **else if** $score(A, B \mid \mathbf{S}) > score(A^\star, B^\star \mid \mathbf{S}^\star)$ **then**
20:       $(A^\star, B^\star \mid \mathbf{S}^\star) \leftarrow (A, B \mid \mathbf{S})$
21: **return** $(A^\star, B^\star \mid \mathbf{S}^\star)$

The algorithm proceeds as follows. In the main loop (line 3) the algorithm iterates over all pairs (A, B) such that $A, B \in \mathbf{V}$ and $A \neq B$. For each such pair it performs a hill-climbing search in the space $\mathbf{2}^{\mathbf{V}-\{A,B\}}$ (lines 8–14) starting with $\mathbf{S} = \varnothing$ and greedily moving to the "neighboring" set of maximum score (loop of lines 12–13), until no such neighbor exists, i.e., until a local optimum is reached (line 14). During this procedure, the set of neighbors Σ_k of a current point $\mathbf{S}$ is defined as all sets $\mathbf{S}' \subseteq \mathbf{V} - \{A, B\}$ that are Hamming distance k from $\mathbf{S}$. The *Hamming distance* between two sets $\mathbf{S}$ and $\mathbf{T}$ is defined as $(|\mathbf{S} - \mathbf{T}| + |\mathbf{T} - \mathbf{S}|)$, where $|\cdot|$ denotes the set cardinality operator. The intuitive explanation of the Hamming distance is that it is equal to the minimum number of simple changes needed to transform set $\mathbf{S}$ into $\mathbf{T}$ (or vice versa), where a "simple change" is the addition or removal of a single variable to or from the set.

The set Σ_k of neighbors of $\mathbf{S}$ that are Hamming distance k away is constructed in line 10. Initially, the hill-climbing procedure is run for $k = 1$, and k is incremented by 1 up to a maximum of K while the initial test (i.e., $(A, B \mid \varnothing)$) is a local maximum. (In our experiments we used $K = 4$.) The rationale for these K repetitions is to improve the chances of getting out of the initial state, when that state is a local maximum. At the end of the iteration for (A, B), the algorithm compares the score of test $(A, B \mid \mathbf{S})$ against $(A^\star, B^\star \mid \mathbf{S}^\star)$, the optimal test so far, and chooses the higher-scoring one. Therefore, at the end of the main loop, $(A^\star, B^\star \mid \mathbf{S}^\star)$ is optimal among all pairs, and the algorithm returns this triplet as the optimal one.

4.4.2   Score Function

We now define the score function given as input to the test optimization algorithm. The main idea is to select the test that brings us closer to the termination condition of zero entropy. We therefore define the score of a candidate test Y_{t+1} as follows:

$$score_t(Y_{t+1}) = -\frac{H_t(X \mid Y_{1:t}, Y_{t+1})}{[W(Y_{t+1})]^\beta} \tag{7}$$

11

where the factor $W(Y)$ denotes the cost of Y and β is a constant parameter provided by the user. The factor $W(Y)^\beta$ in Eq. (12) is used to discourage expensive tests. The *cost of a test* is proportional to the number of variables involved in the test. This is a more realistic assessment of the true computational cost than a simple count of the tests evaluated and is justified as follows. As a statistical test conceptually constructs a contingency table of counts, one for each combination of values of the variables involved in the test, a naïve implementation of such a test would have a cost exponential in the number of variables involved. However, empty cells (containing a count of zero) in this contingency table do not really need to be explicitly represented or enumerated. A more efficient representation of the contingency table therefore is possible—for example, using a sparse representation that does not explicitly store zero counts, possibly implemented using a hash table. The construction of such a data structure therefore only needs to examine each data point once, incrementing the appropriate count of the contingency table. Using such an implementation, the time complexity of a statistical test is proportional to the size of data set $|D|$ and the number of variables involved. For example, a conditional test between variables 1 and 2 given $\{4, 5\}$ has time complexity proportional to $4|D|$. Therefore, in all our time complexity results, we report the weighted number of tests (and not simply the number of tests conducted), referred to as *weighted cost* here on, with the weight of each test conducted equal to the number of variables involved in it. This is a more accurate measure of the actual time that the algorithm will take to execute.

The conditional posterior entropy in the numerator of Eq. (12) is equal to

$$H_t(X \mid Y_{1:t}, Y_{t+1}) = H_t(X \mid Y_{1:t}) - IG_t(Y_{t+1}) \tag{8}$$

where $IG_t(Y_{t+1})$ is the *information gain* of candidate test Y_{t+1}, and can be shown to be equal to:

$$\begin{aligned} IG_t(Y_{t+1}) &= -\sum_x \sum_{y_{1:t}} \Pr_t(x, y_{1:t}) \log \left[\Pr_t(y_{t+1}^x \mid y_{1:t}) \right] \\ &= -\sum_{x \in \mathcal{X}} \Pr_t(x) \log \Pr_t(y_{t+1}^x \mid y_{1:t}^x). \end{aligned} \tag{9}$$

The details of this calculation are presented in Appendix A. The term $\Pr_t(y_{t+1}^x \mid y_{1:t}^x)$ depends solely on $\Pr_t(X)$ as the following calculation demonstrates:

$$\Pr_t(y_{t+1}^x \mid y_{1:t}^x) = \frac{\Pr_t(y_{t+1}^x, y_{1:t}^x)}{\Pr_t(y_{1:t}^x)} = \frac{\sum_{x' \in \{y_{t+1}^x, y_{1:t}^x\}} \Pr_t(x')}{\sum_{x' \in \{y_{1:t}^x\}} \Pr_t(x')} \tag{10}$$

and can therefore be readily estimated using the current population of particles at time t.

Eq. (10) has an interesting and intuitive interpretation: at each time t, the posterior probability of a test Y_{t+1} being true (false), given some assignment $y_{1:t}$ of tests $Y_{1:t}$, equals the fraction of the posterior probability mass at time t of the structures in the equivalence class $\{y_{1:t}\}$ (c.f. Eq.(3)) that are consistent with $Y_{t+1} = \texttt{true}$ ($Y_{t+1} = \texttt{false}$), over the mass of those that exist in class $\{y_{1:t}\}$. Thus, we can estimate it by the fraction of the particles in $\mathcal{X}_t$ in each equivalence class of $Y_{1:t}$ that have Y_{t+1} equal to $\texttt{true}$ or $\texttt{false}$. Note that, under this interpretation, it is not hard to prove that information gain is maximized by a test that splits evenly the mass of every equivalence class generated by the set of tests $Y_{1:t}$ conducted so far.

In the next section we describe in detail the intricacies involved in maintaining a population of particle structures that adequately represent our posterior probability over structures at each step of the algorithm.

12

Algorithm 3 Particle filter algorithm. $\mathcal{X}'' = PF(\mathcal{X}, M, f, q)$

1: $\mathrm{Pr}_t(X) \longleftarrow f(X)$
2: **for all** particles $x \in \mathcal{X}$ **do**
3:     $w(x) \longleftarrow \mathrm{Pr}_t(x)/\mathrm{Pr}_{t-1}(x) \propto \mathrm{Pr}(d_t|y_t^x)$ (see Eq. (4))   /* Compute weights. */
4: **for all** particles $x \in \mathcal{X}$ **do**
5:     $\widetilde{w}(x) \longleftarrow \frac{w(x)}{\sum_{x' \in \mathcal{X}} w(x')}$ /* Normalize weights. */
6: $\mathcal{X}' \longleftarrow resample(\mathcal{X}, \widetilde{w})$ /* Resample particles using residual resampling and $\widetilde{w}$ as the weights. */
7: /* Move each particle in $\mathcal{X}'$ M times. */
8: $\mathcal{X}'' \longleftarrow \varnothing$
9: **for all** $x \in \mathcal{X}'$ **do**
10:     **for** $m = 1$ to M **do**
11:         /* Move particle x using Metropolis-Hastings, distribution $\mathrm{Pr}_t(X)$ and proposal $q(X^\star \mid X)$. */
12:         $x \longleftarrow M\text{-}H(x, \mathrm{Pr}_t, q)$
13:         **if** the proposed move of each particle has been accepted at least π times **then goto** 14
14:     $\mathcal{X}'' \longleftarrow \mathcal{X}'' \cup \{x\}$
15: **return** $\mathcal{X}''$

4.4.3 Particle Filter Implementation for Structures

In this section we explain how the particle filter algorithm can be used to maintain a particle population that represents a posterior probability over structures, such as the one used in PFMN. The particle filter algorithm is shown in Algorithm 3. It is used in line 9 of PFMN to transform population $\mathcal{X}_t$ into $\mathcal{X}_{t+1}$. This is done in a sequence of two steps, *resampling* (lines 2–6) and *moves* (lines 10–13), that are repeated M times in a loop. (In our experiments we used $M = 20$.)

At the beginning of step t, the particle population is distributed according to $\mathrm{Pr}_{t-1}(X)$. However, the correct distribution of interest after observing the result of test Y_t is $\mathrm{Pr}_t(X)$. To adjust for the change in the distribution from $\mathrm{Pr}_{t-1}(X)$ to $\mathrm{Pr}_t(X)$ after each test Y_t a "weight" is attached to each particle. Intuitively, the weight $\mathrm{Pr}_t(x)/\mathrm{Pr}_{t-1}(x) \propto \mathrm{Pr}(d_t|y_t^x)$ represents the number of (possibly fractional) structures represented by each particle x, and is used to compensate for the difference between the sampling and target distributions. (This is an instance of *importance sampling* with importance distribution $\mathrm{Pr}_{t-1}(X)$.) This weighting may result in degeneracies i.e., particles that have very low weight, a problem that may be exacerbated at each new iteration. Resampling is therefore used to eliminate such particles and create new particles at regions of high probability. The normalized weights $\widetilde{w}(x)$ (line 5) are used for this resampling step (line 6). A number of resampling algorithms are available, such as the standard multinomial resampling (Gordon et al., 1993) as well as variance-reducing techniques such as stratified resampling (Doucet et al., 2001; Kitagawa, 1996), systematic resampling (Carpenter et al., 1999) and residual resampling (Liu and Chen, 1998). (In our experiments we used residual resampling.) Finally, the particles are "moved" to produce a more diverse particle population from the current distribution.

As mentioned, during the move phase, all particles are moved (lines 10–13) through a pair of proposal-acceptance steps using the Metropolis-Hastings algorithm (***M-H***), a Markov Chain Monte Carlo (MCMC) algorithm (this approach was first proposed in the context of SMC by Gilks and Berzuini (2001)). The *M-H* algorithm requires that the distribution over structures $\mathrm{Pr}_t(X)$ and a *proposal* distribution $q(X^\star \mid X)$ be provided as parameters. The calculation of the value of $\mathrm{Pr}_t(X)$ has already been previously addressed in Eq. (4); the proposal distribution is discussed in the next paragraph below. Then, given as input an initial particle X and an integer M the algorithm proceeds through M iterations, *proposing* at each iteration a new particle $X^\star$ and *accepting* the proposed particle with probability $\min\left\{1, \frac{\mathrm{Pr}_t(X^\star)}{\mathrm{Pr}_t(X)}\right\}$ (occurring within *M-H*, called in line 12). If

accepted, X is replaced by $X^\star$ before the next iteration, otherwise X remains unchanged. We considered a variation of this standard *M-H* algorithm suggested by Chopin (2002) (section 4.1 titled "*How to move?*") that recommends an early-stop criterion for the standard *M-H* algorithm based on the *acceptance rate* as a measure of *rejuvenation* of the particles. We implemented this by interrupting the propose-accept loop of *M-H* early (line 13), if the proposed move of each particle was accepted at least π times. We used $\pi = 2$ in all our experiments.

We now explain the proposal distribution used in our algorithm. We used a "random walk" proposal $q(x^\star \mid x)$ that generates a sample $x^\star$ from x by iteratively "inverting" each edge of structure x with probability α i.e., changing an edge between each pair of variables (X, Y) to a non-edge or vice versa with probability α. Thus, if h denotes the Hamming distance between structures x and $x^\star$, and $m = n(n-1)/2$, where n is the number of nodes (same for both x and $x^\star$), we have that $q(x^\star \mid x) = q(x \mid x^\star) = \alpha^h (1-\alpha)^{m-h}$. With this proposal, the probability of inverting h edges follows a binomial distribution, with expected number of inversions equal to αm. In our experiments, we use a value for α such that on average, only one edge is inverted per move i.e., $\alpha = 1/m$.

A well-designed proposal $q(X^\star \mid X)$ i.e., one that ensures convergence to the posterior distribution $\Pr_t(X)$, must satisfy the requirements of aperiodicity and irreducibility (Andrieu et al., 2003). The above proposal q satisfies both requirements: it is aperiodic since it always allows for rejection, and irreducible since its support is the entire set of structures $\mathcal{X}$.

5 Experimental Evaluation

In this section we aim to demonstrate the viability of PFMN as an alternative to the GSIMN and GSMN algorithms (Bromberg et al., 2006) when tests are expensive, by showing that it requires a smaller weighted number of tests to learn the networks of Markov networks while maintaining a similar accuracy. We conducted two sets of experiments. In the first set, our experiments were on domains for which the true model was known and generated randomly. In the second set the experiments were on real-world and benchmark data sets, for which the underlying true model was unknown. Our results, presented below, indicate that PFMN outperforms both GSMN and GSIMN while producing networks with comparable accuracy, with few exceptions. Results for the known-model and real-world experiments are described below in Sections 5.1 and 5.2, respectively.

In all experiments we report the *weighted number of tests* and their *accuracy*. The weight of each test is used to account for the execution times of tests with different conditioning set sizes, and is taken to be the number of variables involved in each test (see discussion in Section 4.4.2). To obtain the quality of each recovered network, we measure accuracy as the fraction of *unseen* conditional independence tests that are correct, i.e., we compare the result of each test on the resulting structure (using vertex separation) with the value of the test on the true network or on a very large data set generated from the underlying model (Markov network), corresponding to the approximate limiting value of the test in the network. In essence, this definition of accuracy compares how likely it is for the algorithm to reconstruct the independence assumptions that are present in the few samples provided to it to those of the same algorithm given many samples drawn from the same distribution.

We used the Bayesian independence test of Margaritis (2005) for all three algorithms. Also, in all known-model experiments we considered the following values for parameters of PFMN: $M = 20$ Metropolis-Hasting maximum number of iterations with $\pi = 2$ for the early stop criteria (c.f. Algorithm 3)), $\alpha = 1/\binom{n}{2}$ for the probability of edge reversal in the proposal distribution (as explained earlier in Section 4.4.3, this value of α was chosen so that on average the state of a single

edge is inverted per move), and $\beta = 2$ for the penalizing factor of the score (c.f. Eq. (12)). The remaining parameters vary for different experiments.

5.1 Known-Model Experiments

We first evaluated our algorithm in artificial domains in which the structure of the underlying model, called the *true network*, is known. This allowed a systematic study of its behavior under varying network densities (reflected in the number of edges in the true network). True networks contained $m_\tau = \tau \frac{n}{2}$ edges, where τ is called the *connectivity* parameter and equals the average number of neighbors of a node (averaged over all nodes in the network), a factor that represents the density of the network. The edges of all true networks used were generated randomly by selecting the first $m_\tau = \tau \frac{n}{2}$ pairs in a random permutation of all possible edges. The factor $1/2$ is necessary because each edge contributes to the degree of two nodes.

Two types of known-model experiments were conducted. In the first set, information on underlying independences was obtained by vertex separation on the true network and thus is 100% accurate. Results of these experiments are presented in the next section. In the second type of experiments, information of the underlying independences was obtained through statistical tests performed on data sampled from the true network using a Gibbs sampler (Geman and Geman, 1984).

All networks used in the known-model experiments have a domain size of $n = 20$.
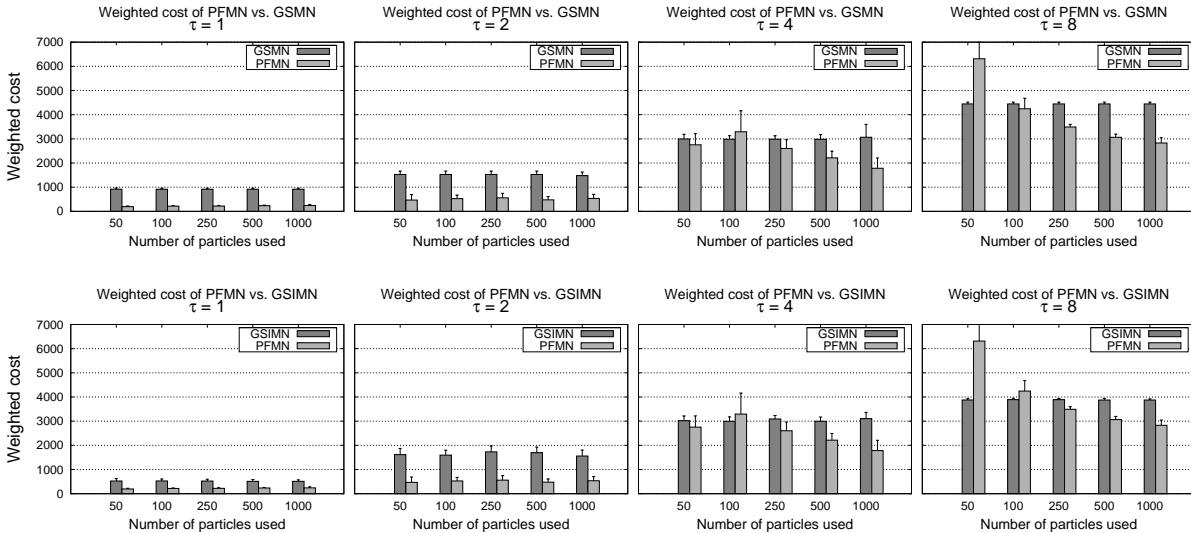
5.1.1 Exact Learning Experiments

Figure 3: Comparison of the weighted number of tests required by PFMN vs. GSMN (top row) and PFMN vs. GSIMN (bottom row) to learn a Markov network in the exact learning case, for varying number of particles $N = 50, 100, 250, 500,$ and 1000 and varying connectivity parameter $\tau = 1, 2, 4$ and 8.

Figures 3 and 4 show the results of the exact learning experiments, where the ground truth comes from conditional independence tests conducted directly on the true network through vertex separation i.e., the outcomes of tests are always correct. Figure 3 shows the absolute value of the weighted number of tests conducted by PFMN compared to both GSMN and GSIMN for a number
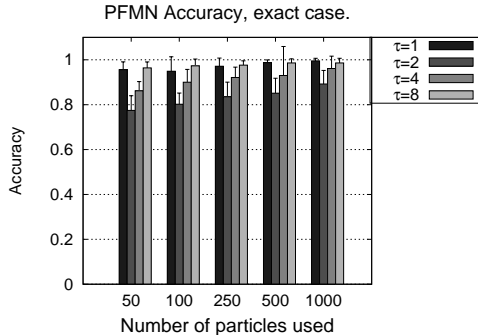
PFMN Accuracy, exact case.

Figure 4: Accuracy of the Markov networks output by PFMN in the exact learning case for varying number of particles $N = 50, 100, 250, 500,$ and 1000 and varying connectivity parameters $\tau = 1, 2, 4,$ and 8.

of connectivities ($\tau = 1, 2, 4,$ and 8), and an increasing number of structure particles N. Ten true networks were generated randomly for each value of τ. The bars shows the mean value and the error bars the standard deviation.

We can see in Figure 3 that for large N the weighted cost of PFMN tends to be lower than that of GSIMN and GSMN for all connectivities, reaching savings of up to 75% for $\tau = 1$. The cases of $\tau = 4$ and $\tau = 8$ shows the difficulty of PFMN to perform better than GSIMN and GSMN for small N. The performance increases for large N, reaching improvements of almost 50% for $\tau = 4$ for both competitors GSMN and GSIMN.

Contrary to GSMN and GSIMN, PFMN is an approximate algorithm, and thus even for the case of exact tests it may produce a network different than the true one. We thus measure the estimated accuracy of the networks produced by PFMN. This accuracy was calculated using the following equation:

$$\widehat{acc}_{PFMN} = \frac{1}{|\mathcal{T}|}\left|\left\{t \in \mathcal{T} \mid I_{PFMN}(t) = I_{true}(t)\right\}\right|. \tag{11}$$

where $t \in \mathcal{T}$ denotes a test, $I_{true}(t)$ denotes the result of test t performed on the true network, and $I_{PFMN}(t)$ denotes the result of test t performed on the output network produced by PFMN. This accuracy definition compares the result (`true` or `false`) of a number of conditional independence tests on the output network (using vertex separation) to the same tests performed on the true network (also using vertex separation). For this calculation, we sampled randomly a set $\mathcal{T}$ of 2000 triplets $(A, B, \mathbf{S})$ evenly distributed among all possible conditioning set sizes $m \in \{0, \ldots, n-2\}$ (i.e., $2000/(n-1)$ tests for each m). Each of these triplets was constructed as follows: First, two variables A and B were drawn randomly from $\mathbf{V}$. Second, the conditioning set was determined by picking the first m variables from a random permutation of $\mathbf{V} - \{A, B\}$. This ensures a uniform distribution over all possible triplets of each size. Figure 4 depicts accuracy results for PFMN. The figure shows that even though the accuracies $\widehat{acc}_{PFMN}$ of PFMN are not 1, in most cases they show an increasing trend for large values of N for sparser domains ($\tau = 1, 2$).

5.1.2 Sampled Data Experiments

We also evaluated PFMN, GSMN, and GSIMN on data sets sampled from the same known networks as the previous section using Gibbs sampling. To sample data from a Markov network, in addition to the network structure, the network parameters (clique potentials) need to be specified. For any given network, its parameters determine the strength of dependencies among connected variables in
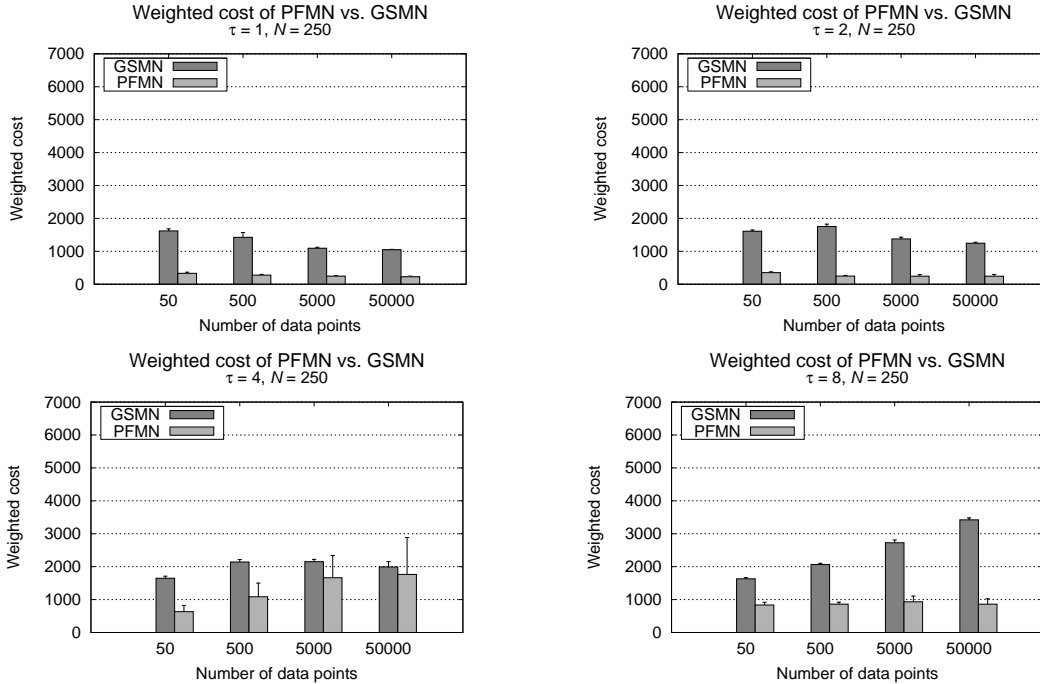
Figure 5: Weighted cost comparison of PFMN vs. GSMN for sampled data sets with increasing number of data points used. The x-axis is plotted in log-scale. The figure shows four plots for a number of connectivities ($\tau = 1, 2, 4$, and 8), for a fixed number of particles $N = 250$.

the graph. Following Agresti (2002), the strength of the probabilistic influence between two binary variables A and B can be measured by the *log-odds ratio* defined as

$$\theta_{XY} = \log \frac{\Pr(A = 0, B = 0) \Pr(A = 1, B = 1)}{\Pr(A = 0, B = 1) \Pr(A = 1, B = 0)}.$$

We generated the network parameters randomly in a way such that the log-odds ratio between every pair of variables connected by an edge in the graph has a specified value. In this set of experiments, we used $\theta = 1$ for every such pair.

Tests conducted on data may be inaccurate, i.e., the independences they produce may not match the independences in the true network. As a consequence, the accuracy of the networks learned by PFMN, GSMN, and GSIMN, are not guaranteed to match the true network. This accuracy is expected, however, to improve with larger data sets. We thus report, in addition to comparisons on the weighted cost of the three algorithms, the accuracy of the networks they produce on increasing data set sizes. Since these experiments are conducted on data sampled from a known network we can compute their accuracy by comparing the output network with true network using Eq. (11) as done for the exact learning experiments of the previous section.

Figure 5 shows the weighted number of tests of PFMN vs. GSMN for a number of connectivities ($\tau = 1, 2, 4$, and 8), for a fixed number of particles $N = 250$ and data sets containing increasing number of data points. The x-axis is plotted using log-scale. Again, ten true networks were generated randomly for each τ, and the figure shows the mean value of the weighted cost using histogram bars, and the standard deviation using 95% error bars. In all cases with the exception of $\tau = 4$ (large number of data points), PFMN demonstrates a considerable reduction in the weighted number of tests, reaching statistically significant savings of up to 90% for $\tau = 2$, $|d| = 500$.
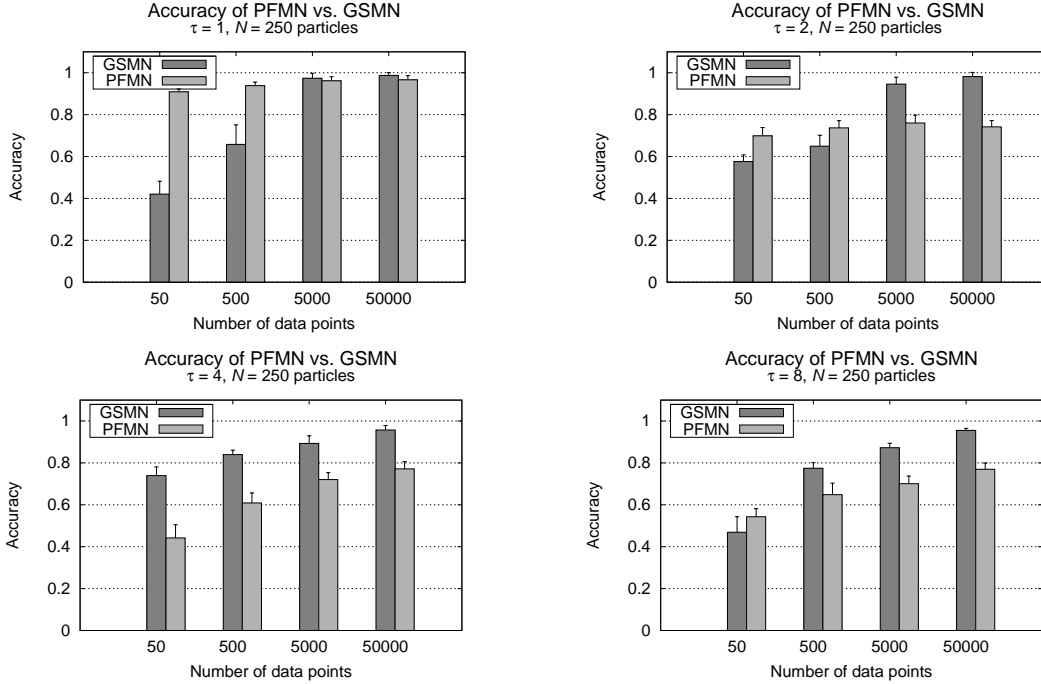
Figure 6: Accuracy comparison of PFMN vs. GSMN for sampled data sets with increasing number of data points used. The x-axis is plotted in log-scale. The figure shows four plots for a number of connectivities ($\tau = 1, 2, 4$, and 8) and a fixed number of particles $N = 250$.

This indicates that the use of the PFMN algorithm can be beneficial in cases where the domain's connectivity is moderate. The case of $\tau = 4$ illustrates the difficulty of PFMN to perform better than GSMN. This can be explained by the fact that the number of structures grows exponentially with the connectivity τ as the number of edges $m_\tau = \tau \frac{n}{2}$ in the network approach $\frac{1}{2}\binom{n}{2}$ (from either side), i.e., the middle point between an empty and a completely connected network, making the structure search task of PFMN more difficult.

The estimated accuracies of PFMN vs. GSMN (i.e., $\widehat{acc}_{PFMN}$ and $\widehat{acc}_{GSMN}$ respectively, where $\widehat{acc}_{GSMN}$ is defined in a way analogous to Eq. (11)) are shown in Figure 6 for the same set of connectivity parameters $\tau = 1, 2, 4, 8$. In all eight plots, the mean value and standard deviation over ten runs of the absolute accuracy are shown. Again, the x-axis is plotted using log-scale. The results in this figure are mixed, showing comparable accuracies between PFMN and GSMN for $\tau = 1$, a shortening gap as d increases for the case of $\tau = 4$, and decline in accuracy of PFMN in the other cases.

Figures 7 and 8 show the results of the same set of experiments but comparing PFMN to GSIMN instead of GSMN. Figure 7 shows the weighted number of tests for connectivities $\tau = 1, 2, 4$, and 8. Once again, the figure shows the mean values and standard deviation over the same group of random networks generated over each value of τ. Qualitatively the results are similar to the comparison against GSMN but this time with smaller difference between the two weighted costs. This is expected because GSIMN typically outperforms GSMN in the weighted number of tests required (Bromberg et al., 2006). In all cases, with the exception of $\tau = 4$, PFMN demonstrates a considerable reduction in the weighted number of tests, reaching savings of up to 60% for $\tau = 2$, $|d| = 50000$. Figure 8 shows the accuracy results. The results in this figure indicate that the accuracy of PFMN is comparable to that of GSIMN for large $|d|$, practically matching accuracies
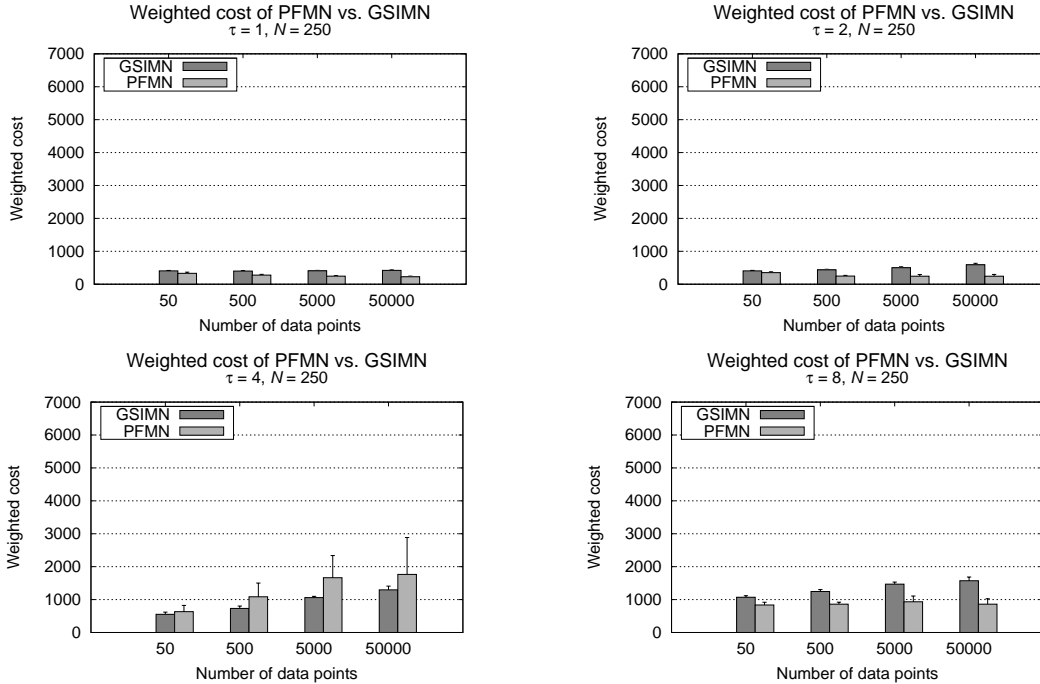
Figure 7: Weighted cost comparison of PFMN vs. GSIMN for sampled data sets with increasing number of data points used. The x-axis is plotted in log-scale. The figure shows four plots for a number of connectivities ($\tau = 1, 2, 4$, and 8), and a fixed number of particles $N = 250$.

(slightly better for PFMN) for $\tau = 1$, with differences in accuracy of less than 5% for $\tau = 4$, increasing to differences of up to 10% in the other two cases.

5.2   Real-World and Benchmark Data Experiments

While known-model experiments have the advantage of allowing a more controlled and systematic study of their performance, experiments on real-world data are necessary for a more realistic assessment of their performance. Real data sets are more challenging because their underlying probability distribution may not be faithful to any Markov network, and its structure may be non-random (e.g., a possibly irregular lattice in many cases of spatial data). We therefore also conducted experiments on a number of data sets obtained from the UCI Machine Learning (Newman et al., 1998) and KDD repositories (Hettich and Bay, 1999).

As above, we compare the weighted cost and accuracy of PFMN vs. GSMN and GSIMN. However, the underlying model is unknown in this experiments. We thus learn the model (in all three algorithms) using a small fraction of the data points (1/3) as input, and report the accuracy that results from comparing the outcome of independence tests conducted on the network output by the algorithms and independence tests conducted on the complete data set. As in the exact learning experiments of the previous section, we used a set $\mathcal{T}$ of 2000 randomly generated triplets and, for each triplet $t \in \mathcal{T}$, compared the result of conducting a statistical conditional independence test for t on the data set containing all possible data points, denoted $I_{data}(t)$, to the outcome of the test performed on through vertex-separation on the network output by each of the algorithms, denoted $I_{ALG}(t)$, $ALG \in \{\text{PFMN}, \text{GSMN}, \text{GSIMN}\}$ respectively. Our reported accuracy was computed as
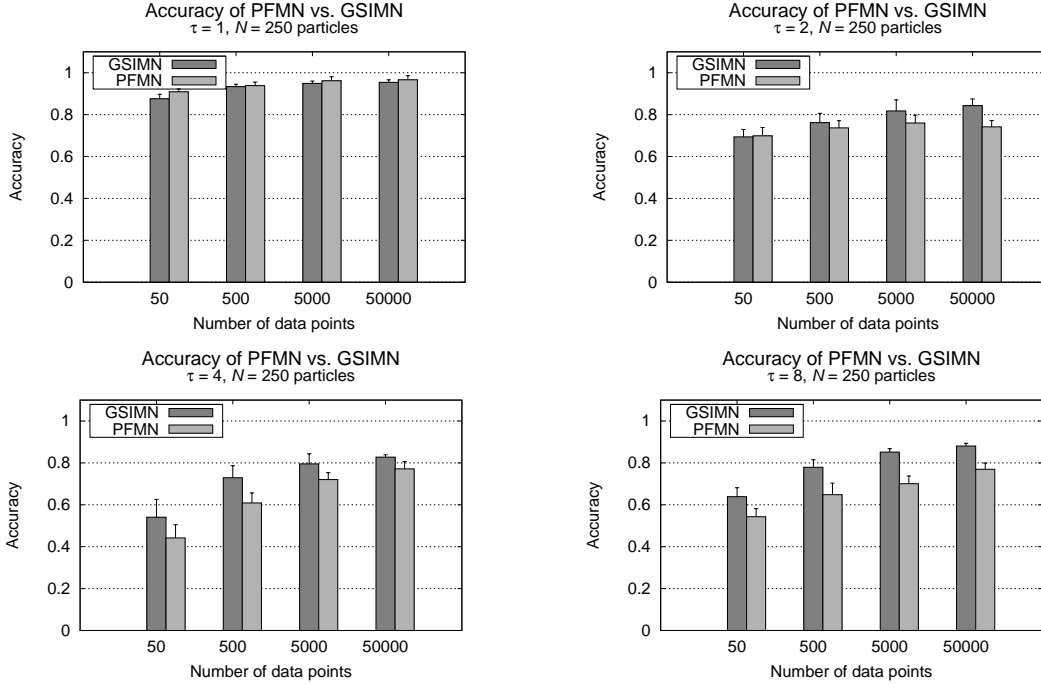
Figure 8: Accuracy comparison of PFMN vs. GSIMN for sampled data sets with increasing number of data points used. The x-axis is plotted in log-scale. The figure shows eight plots for a number of connectivities ($\tau = 1, 2, 4,$ and 8), and a fixed number of particles $N = 250$.

follows:

$$\widehat{acc}_{ALG} = \frac{1}{|\mathcal{T}|}\left|\left\{t \in \mathcal{T} \mid I_{ALG}(t) = I_{data}(t)\right\}\right|.$$

Table 1 shows the results in detail, highlighting (in bold font) the best performance among GSMN, GSIMN, and PFMN. These results are also plotted in Figure 9 (left plot). The figure shows the ratio of the weighted cost of PFMN vs. GSMN (with a number smaller than 1 shows improvement of PFMN vs. GSMN), and the difference in the accuracies of PFMN and GSMN (with a positive histogram bar shows an improvement of PFMN vs. GSMN). In these experiments, PFMN used $N = 2n$ to adjust for the effect of varying numbers of variables across domains. The numbers in the x-axis correspond to indices to the data sets shown in Table 1. In all 16 data sets PFMN required lower weighted cost than GSMN, reaching ratios as low as 0.12 (i.e., a 88% reduction). Moreover, for 10 out of 16 data sets PFMN achieves this reduction in cost with better or no reduction in accuracy compared to GSMN, reaching improvements of up to 17% for the nursery and tic-tac-toe data sets (indexes 7 and 9, respectively). The rest exhibit only a modest reduction of less than 6.8% (the largest difference occurring for the "hayes-roth" data set).

Figure 9 (right plot) shows a graph similar to Figure 9 (left plot) but comparing GSIMN vs. PFMN instead. As in the experiments for GSMN, PFMN used $N = 2n$, and the numbers in the x-axis are indices to the data sets shown in Table 1. We can see that in 13 out of 16 data sets PFMN required lower weighted cost than GSIMN, reaching ratios as low as 0.38 (i.e., a 62% reduction). As in the GSMN case, in many cases (8 out of 16) PFMN achieves this reduction in cost with better or no reduction in accuracy compared to GSIMN. The remaining cases exhibit only a modest reduction of less than 3.8% with the exception of the "balance-scale" data set showing a reduction of 6.6% and the "baloons" data set showing a reduction of 13.2%.
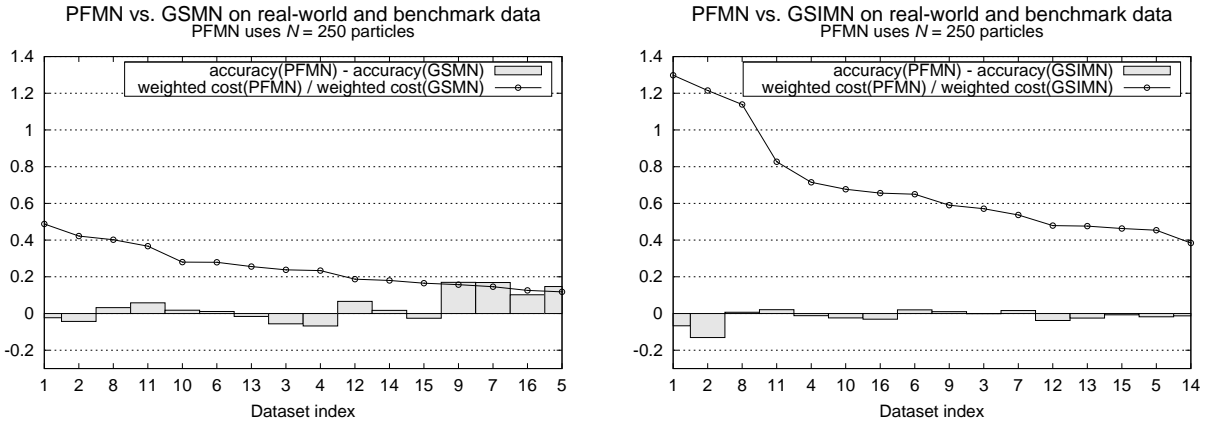
20

Figure 9: Performance comparison of PFMN vs. GSMN (left plot) and GSIMN (right plot) on real-world and benchmark data sets. The line curve shows the ratio of weighted cost of PFMN vs. GSMN or GSIMN, while the histogram bars show the difference between accuracy of PFMN and GSMN or GSIMN on real-world data sets. The numbers on the x-axis correspond to indices to data set as listed in Table 1.

6   Conclusion

We presented PFMN, an independence-based algorithm for learning the structure of the Markov network of a domain from data. We also presented, for the first time to our knowledge, a generative model that involves structures, tests and data that explicitly and precisely describes the interactions among these random quantities. We also provided a simplified model that allows the analytical evaluation of the posterior probability over structures, and presented a theoretical analysis and computation of the posterior, entropy and expected information gain of any test under the simplified generative model. We also presented a practical algorithm (PFMN) and practical implementation based on particle filters that greedily chooses, at each step, the most informative test (according to information gain) to evaluate, thus minimizing the cost of tests performed during its execution. We compared PFMN to existing approaches and showed experimentally that it executes fewer weighted tests in sparse or moderately connected domains with similar levels of accuracy. It is therefore a useful approach in cases where independence tests are expensive, such when data sets are very large and/or distributed over a potentially slow network, making the execution of each test expensive.

A   Calculation of Information Gain of Test Y_{t+1}

As mentioned in Section 4.4.2, the score of a candidate test Y_{t+1} used in the optimization algorithm (Algorithm 2) is

$$score_t(Y_{t+1}) = -\frac{H_t(X \mid Y_{1:t}, Y_{t+1})}{[W(Y_{t+1})]^\beta} \tag{12}$$

where the factor $W(Y)$ denotes the cost of Y and β is a constant parameter provided by the user. In this appendix we calculate the conditional posterior entropy in the numerator of Eq. (12).

Let us consider a candidate test Y_{t+1}. $H_t(X \mid Y_{1:t}, Y_{t+1})$ is the entropy over structures given the (not yet performed) test Y_{t+1}, and is equal to:

$$H_t(X \mid Y_{1:t}, Y_{t+1}) = H_t(X \mid Y_{1:t}) - IG_t(Y_{t+1}). \tag{13}$$

Table 1: Comparison of weighted number of tests and accuracy of PFMN, GSMN, and GSIMN for several real-world and benchmark data sets. For each evaluation measure, the best performing algorithm is indicated in bold. The table also reports n, the number of variables in the domain, and N, the number of data points in each data set.

Data set					Weighted #(tests)			Accuracy		
#	name	n	$|D|$	N	PFMN	GSMN	GSIMN	PFMN	GSMN	GSIMN
1	balance-scale	5	211	10	29.429	61.143	**23.429**	0.663	0.686	**0.729**
2	baloons	5	7	10	26.857	64.429	**22.571**	0.712	0.755	**0.844**
3	haberman	5	111	10	**11.429**	48	20	0.656	**0.712**	0.657
4	hayes-roth	6	48	12	**26.429**	112.143	37.286	0.705	**0.773**	0.716
5	car	7	598	14	**20.286**	172.429	45.857	0.764	0.617	**0.781**
6	monks-1	7	187	14	**27.286**	97.571	42	**0.920**	0.909	0.901
7	nursery	9	4331	18	**50**	342.143	93.143	**0.762**	0.593	0.746
8	cmc	10	503	20	161.429	404.714	**142**	**0.693**	0.661	0.686
9	tic-tac-toe	10	315	20	**60.429**	384.714	106.429	**0.694**	0.525	0.684
10	bridges	12	27	24	**127.857**	453.571	189.143	0.697	0.679	**0.721**
11	crx	16	218	32	**395.714**	1084.286	476.857	**0.728**	0.669	0.707
12	hepatitis	20	31	40	**245.571**	1332.714	516.143	0.777	0.711	**0.815**
13	imports-85	25	67	50	**599.875**	2341.750	1266.500	0.429	0.445	**0.454**
14	flag	29	67	58	**610.143**	3417.857	1597.143	0.484	0.467	**0.496**
15	dermatology	35	126	70	**1134.143**	6888.429	2478.714	0.351	**0.377**	0.358
16	alarm	37	6692	74	**1347.714**	10622.571	2080.714	0.535	0.433	**0.567**

In this expression $IG_t(Y_{t+1})$ denotes the information gain of candidate test Y_{t+1}. The derivation of $IG_t(Y_{t+1})$ is as follows. By the definition of conditional entropy we have that:

$$
\begin{aligned}
& H_t(X \mid Y_{1:t}, Y_{t+1}) \\
&= -\sum_x \sum_{y_{1:t}} \sum_{y_{t+1}} \mathrm{Pr}_t(x, y_{1:t}, y_{t+1}) \log\left[\mathrm{Pr}(x \mid y_{1:t}, y_{t+1})\right] \\
&= -\sum_x \sum_{y_{1:t}} \sum_{y_{t+1}} \mathrm{Pr}_t(y_{t+1} \mid x, y_{1:t}) \mathrm{Pr}_t(x, y_{1:t}) \log\left[\frac{\mathrm{Pr}(y_{t+1} \mid y_{1:t}, x) \, \mathrm{Pr}(x \mid y_{1:t})}{\mathrm{Pr}_t(y_{t+1} \mid y_{1:t})}\right].
\end{aligned}
\tag{14}
$$

Given our generative model, $\mathrm{Pr}(y_{t+1} \mid x, y_{1:t}) = \mathrm{Pr}(y_{t+1} \mid x)$ which, according to Eq. (2), equals $\delta(y_{t+1}, y_{t+1}^x)$. Therefore, the above becomes

$$
\begin{aligned}
& H_t(X \mid Y_{1:t+1}) \\
&= -\sum_x \sum_{y_{1:t}} \mathrm{Pr}_t(x, y_{1:t}) \log\left[\frac{\mathrm{Pr}_t(x \mid y_{1:t})}{\mathrm{Pr}_t(y_{t+1}^x \mid y_{1:t})}\right] \tag{15} \\
&= -\sum_x \sum_{y_{1:t}} \mathrm{Pr}_t(x, y_{1:t}) \log\left[\mathrm{Pr}_t(x \mid y_{1:t})\right] + \sum_x \sum_{y_{1:t}} \mathrm{Pr}_t(x, y_{1:t}) \log\left[\mathrm{Pr}_t(y_{t+1}^x \mid y_{1:t})\right]. \\
&= H_t(X \mid Y_{1:t}) + \sum_x \sum_{y_{1:t}} \mathrm{Pr}_t(x, y_{1:t}) \log\left[\mathrm{Pr}_t(y_{t+1}^x \mid y_{1:t})\right]. \tag{16}
\end{aligned}
$$

Comparing this to Eq. (13), we see that the second term is (the negative of) the information gain

of candidate test Y_{t+1} given all information at time t i.e.,

$$
\begin{aligned}
IG_t(Y_{t+1}) &= -\sum_x \sum_{y_{1:t}} \Pr_t(x, y_{1:t}) \log \left[\Pr_t(y_{t+1}^x \mid y_{1:t}) \right] \\
&= -\sum_{x \in \mathcal{X}} \Pr_t(x) \log \Pr_t(y_{t+1}^x \mid y_{1:t}^x). \quad (17)
\end{aligned}
$$

since $\Pr_t(x, y_{1:t}) = \Pr_t(y_{1:t} \mid x)\Pr_t(x) = \delta(y_{t+1}, y_{t+1}^x)\Pr_t(x)$ and therefore only the term $y_{1:t} = y_{1:t}^x$ survives in the summation over tests, all others being equal to zero according to Eq. (2).

As with all other quantities that requires a summation over the space of particles $\mathcal{X}$, we can approximate the information gain as computed in Eq. (17) using Eq. (6), i.e.,

$$
IG_t(Y_{t+1}) \approx -\frac{1}{N} \sum_{i=1}^N \log f(\Pr_t(x^{(i)}))
$$

where $f(\Pr_t(x))$ denotes the quantity $\Pr_t(y_{t+1}^x \mid y_{1:t}^x)$.

B Bayesian Conditional Independence Likelihood Calculation

In this appendix we briefly describe the calculation of the likelihoods given the value of a statistical test (independence and dependence) used by the PFMN algorithm to update the posterior distributions after each test in the sequence is done (lines 6 and 7 in Algorithm 1). We follow the Bayesian test of Margaritis (2005), adapted to the case of discrete variables. We first describe the unconditional case, followed by the conditional test that makes use of the unconditional version. The unconditional Bayesian independence test first appeared in Margaritis and Thrun (2001).

The central idea of the test between X and Y is the comparison of two competing statistical models, $M_{\mathcal{I}}$ (the *independent* model) and $M_{\neg \mathcal{I}}$ (the *dependent* model), according to their posterior probability given a data set consisting of (X, Y) pairs. As the PFMN algorithm only needs their likelihoods, we only present that here. The dependent model $M_{\neg \mathcal{I}}$ corresponds to a joint multinomial distribution while the independent model $M_{\mathcal{I}}$ to two marginal multinomials along the X- and Y-axes. Let us assume that the data set d is used to construct an $I \times J$ table of counts $\{c_{ij} \mid 1 \leq i \leq I, 1 \leq j \leq J\}$. The dependent model $M_{\neg \mathcal{I}}$ consists or a multinomial distribution over the counts for the "joint" (X, Y) random variable with parameters $\{p_{ij}\}$, where p_{ij} is the probability of occurrence of event $(X = i, Y = j)$, that is

$$
\Pr(d \mid \{p_{ij}\}, M_{\neg \mathcal{I}}) = M! \prod_{i=1}^I \prod_{j=1}^J \frac{p_{ij}^{c_{ij}}}{c_{ij}!}
$$

where $M = |d|$ is the size of our data set. The parameters $\{p_{ij}\}$ are typically unknown. We therefore adopt a Bayesian approach: we use a prior distribution $\Pr(\{p_{ij}\})$ over them. Given such a distribution the data likelihood is the average over all possible parameter values, weighted by their probability:

$$
\Pr(d \mid M_{\neg \mathcal{I}}) = \int Pr(d \mid \{p_{ij}\}, M_{\neg \mathcal{I}}) \Pr(\{p_{ij}\} \mid M_{\neg \mathcal{I}}) \, dp_{11} \cdots dp_{IJ}. \quad (18)
$$

The most commonly used prior distribution for multinomial parameters is the Dirichlet:

$$
\Pr(\{p_{ij}\} \mid M_{\neg \mathcal{I}}) = \Gamma(\gamma) \prod_{i=1}^I \prod_{j=1}^J \frac{p_{ij}^{\gamma_{ij}-1}}{\Gamma(\gamma_{ij})},
$$

where $\gamma = \sum_{i=1}^{I} \sum_{j=1}^{J} \gamma_{ij}$ and $\Gamma(x)$ is the gamma function.[1] The positive numbers $\{\gamma_{ij}\}$ are the *hyperparameters* of the Dirichlet distribution. When $\gamma_{ij} = 1$ for all i, j, the Dirichlet prior is uniform.

The choice of a Dirichlet prior has certain computational advantages. Since it is conjugate prior to the multinomial, the posterior distribution of the parameters is also Dirichlet (albeit with different parameters). This enables the computation of the integral of Eq. (18) analytically:

$$\Pr(d \mid M_{\neg\mathcal{I}}) = \frac{\Gamma(\gamma)}{\Gamma(\gamma + M)} \prod_{i=1}^{I} \prod_{j=1}^{J} \frac{\Gamma(\gamma_{ij} + c_{ij})}{\Gamma(\gamma_{ij})}.$$

Similar calculations can be made for the independent model $M_{\mathcal{I}}$ which consists of two multinomials with (unknown) parameters $\{q_i\}$ (X-axis) and $\{r_j\}$ (Y-axis), with priors Dirichlet distributions with hyperparameters $\{\alpha_i\}$ and $\{\beta_j\}$, respectively. The probability of each contingency table cell (i, j) in this model is the product of the respective marginals q_i and r_j. Using similar calculations as above, the data likelihood under this model is:

$$\Pr(d \mid M_{\mathcal{I}}) = \left(\frac{\Gamma(\alpha)}{\Gamma(\alpha + M)} \prod_{i=1}^{I} \frac{\Gamma(\alpha_i + c_{i\cdot})}{\Gamma(\alpha_i)} \right) \left(\frac{\Gamma(\beta)}{\Gamma(\beta + M)} \prod_{j=1}^{J} \frac{\Gamma(\beta_j + c_{\cdot j})}{\Gamma(\beta_j)} \right)$$

where $\alpha = \sum \alpha_i$, $\beta = \sum \beta_j$ and $c_{i\cdot} = \sum_{j=1}^{M} c_{ij}$ and $c_{\cdot j} = \sum_{i=1}^{M} c_{ij}$ are the marginal counts along the X- and Y-axes, respectively.

The calculation of the data likelihood of the conditional Bayesian test is similar. A version of this appeared in Margaritis (2005) for continuous variables; here we present a modified version that is appropriate for discrete variables, for conditional test $(X, Y \mid \mathbf{Z})$. Let us assume that the variables is $\mathbf{Z}$ collectively take one of K value combinations (for example, if $\mathbf{Z}$ contains n binary variables then $K = 2^n$). Conditioning on different values $\mathbf{Z} = \mathbf{z}$ partitions the data set d into K disjoint sub-data sets d_k, sometimes called "slices," i.e., $d_i \cap d_j = \varnothing$ for $i \neq j$ and $\bigcup_{k=1}^{K} d_k = d$. The conditional test again uses two competing models, the dependent model $M_{\neg\mathcal{CI}}$ and the independent model $M_{\mathcal{CI}}$. $M_{\mathcal{CI}}$ implies unconditional independence of X and Y at each slice, while $M_{\neg\mathcal{CI}}$ implies unconditional dependence in at least one of the slices. The likelihood of unconditional dependence and independence at each slice k can be calculated as above using d_k as input, i.e., $\Pr(d_k \mid M_{\mathcal{I}}^{k}) \overset{\text{def}}{=} g_k$ and $\Pr(d_k \mid M_{\neg\mathcal{I}}^{k}) \overset{\text{def}}{=} h_k$, where $M_{\mathcal{I}}^{k}$ and $M_{\neg\mathcal{I}}^{k}$ are respectively the independent and dependent model of slice k. As the data at different slices are disjoint and independent, the data likelihood of d under $M_{\mathcal{CI}}$ is

$$\Pr(d \mid M_{\mathcal{CI}}) = \prod_{k=1}^{K} \Pr(d_k \mid M_{\mathcal{I}}^{k}) = \prod_{k=1}^{K} g_k. \tag{19}$$

The data likelihood under $M_{\neg\mathcal{CI}}$ is a little more complex to calculate but still analytically computable; it consists of summing over all possible values of independence and dependence for the slices of the conditional test. Using the definition of conditional dependence again results in the following likelihood:

$$\Pr(d \mid M_{\neg\mathcal{CI}}) = \frac{\prod_{k=1}^{K} (p_k g_k + q_k h_k) - \prod_{k=1}^{K} g_k}{\Pr(M_{\neg\mathcal{CI}})}, \tag{20}$$

where $p_k = \Pr(M_{\mathcal{I}}^{k}) \overset{\text{def}}{=} \Pr(M_{\mathcal{I}})^{1/K}$ and $q_k = 1 - p_k$ are the priors of the independent and dependent models of slice k, respectively.

---

[1]The gamma function is defined as $\Gamma(x) = \int_{0}^{+\infty} e^{-t} t^{x-1} dt$. When x is a non-negative integer, $\Gamma(x+1) = x!$.

References

P. Abbeel, D. Koller, and A. Y. Ng. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, 7:1743–1788, 2006.

A. Agresti. *Categorical Data Analysis*. Wiley, 2nd edition, 2002.

C. F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON, a novel Markov blanket algorithm for optimal variable selection. In *Proceedings of the American Medical Informatics Association (AMIA) Fall Symposium*, 2003.

C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of Markov random fields for segmentation of 3D range data. *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

F. Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, 1982.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B (Methodological)*, 57(1):289–300, 1995.

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29(4):1165–1188, 2001.

J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 1974.

J. Besag, J. York, and A. Mollie. Bayesian image restoration with two applications in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43:1–59, 1991.

F. Bromberg, D. Margaritis, and V. Honavar. Efficient Markov network structure discovery using independence tests. In *SIAM International Conference on Data Mining*, 2006.

O. Cappé, A. Guillin, J.-M. Marin, and C. P. Robert. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929, 2004.

J. Carpenter, P. Clifford, and P. Fearnhead. An improved particle filter for non-linear problems. *IEE proceedings, Radar, Sonar and Navigation*, 146:2–7, 1999.

N. Chopin. A sequential particle filter method for static models. *Biometrika*, 89:539–552, 2002.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462 – 467, May 1968.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society, Series B (Methodological)*, 68:411–436, 2006.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):390–393, 1997.

A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo methods in practice.* Springer Verlag, New York, 2001.

D. Edwards. *Introduction to Graphical Modelling.* Springer, New York, 2nd edition, 2000.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian relation of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

W. R. Gilks and C. Berzuini. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society, Series B (Methodological)*, 63:127–146, 2001.

N. Gordon, D. Salmond, and A. F. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F, Radar and Signal Processing*, 140(2):107–113, 1993.

D. Heckerman. A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

S. Hettich and S. D. Bay. UCI KDD archive. University of California, Irvine, Dept. of Information and Computer Science, 1999.

R. Hofmann and V. Tresp. Nonlinear Markov networks for continuous variables. In *NIPS '98*, volume 10, pages 521–529, 1998.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087–1116, 1993.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

W. Lam and F. Bacchus. Learning Bayesian belief networks: an approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

S.-I. Lee, V. Ganapahthi, and D. Koller. Efficient structure learning of Markov networks using L_1-regularization. In *Neural Information Processing Systems (NIPS)*, 2007.

J. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

D. Margaritis. Distribution-free learning of Bayesian network structure in continuous domains. In *National Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2005.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 505–511. MIT Press, 2000.

D. Margaritis and S. Thrun. A Bayesian multiresolution independence test for continuous variables. In *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 346–353. Morgan Kaufmann, 2001.

A. McCallum. Efficiently inducing features of conditional random fields. *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2003.

R. G. Miller. *Simultaneous Statistical Inference.* Springer Verlag (New York), 2nd edition, 1981.

D. J. Newman, S. Hettich, C. L. Blake, and C. Merz. UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., 1988.

G. Rebane and J. Pearl. The recovery of causal poly-trees from statistical data. In L. N. Kanal, T. S. Levitt, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 175–182, Amsterdam, 1989. North-Holland.

S. Shekhar, P. Zhang, Y. Huang, and R. Vatsavai. Trends in spatial data mining. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, editors, *Data Mining: Next Generation Challenges and Future Directions*, chapter 19, pages 357–379. AAAI Press / The MIT Press, 2004.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* Adaptive Computation and Machine Learning Series. MIT Press, 2nd edition, January 2000.

N. Srebro and D. Karger. Learning Markov networks: Maximum bounded tree-width graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, 2001.

S. Tong and D. Koller. Active learning for structure in Bayesian networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Algorithms for large scale Markov blanket discovery. In *Proceedings of the 16th International FLAIRS Conference*, pages 376–381, 2003a.

I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 673–678, 2003b.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 2006.

J. Whittaker. *Graphical Models in Applied Multivariate Statistics.* John Wiley & Sons, New York, 1990.