In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI), 2007.

Efficient and Robust Independence-Based Markov Network Structure Discovery

Facundo Bromberg and **Dimitris Margaritis**
Department of Computer Science
Iowa State University
Ames, IA 50011

Abstract

In this paper we introduce a novel algorithm for the induction of the Markov network structure of a domain from the outcome of conditional independence tests on data. Such algorithms work by successively restricting the set of possible structures until there is only a single structure consistent with the conditional independence tests executed. Existing independence-based algorithms have well-known shortcomings, such as rigidly ordering the sequence of tests they perform, resulting in potential inefficiencies in the number of tests required, and committing fully to the test outcomes, resulting in lack of robustness in case of unreliable tests. We address both problems through a Bayesian particle filtering approach, which uses a population of Markov network structures to maintain the posterior probability distribution over them, given the outcomes of the tests performed. Instead of a fixed ordering, our approach greedily selects, at each step, the optimally informative from a pool of candidate tests according to information gain. In addition, it maintains multiple candidate structures weighed by posterior probability, which makes it more robust to errors in the test outcomes. The result is an approximate algorithm (due to the use of particle filtering) that is useful in domains where independence tests are uncertain (such as applications where little data is available) or expensive (such as cases of very large data sets and/or distributed data).

1 Introduction

In this paper we focus on the task of learning the structure of **Markov networks** (**MNs**), a subclass of graphical models, from data in discrete domains. (Other graphical models include Bayesian networks, represented by directed graphs.) MNs consist of two parts: an undirected graph (the model structure), and a set of parameters. An example Markov network is shown in Fig. 1. 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.

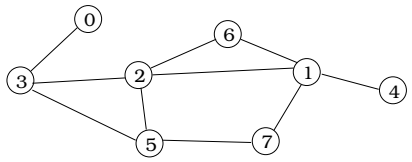

Figure 1: Example Markov network. The nodes represent variables in the domain $\mathcal{V} = \{0, 1, 2, 3, 4, 5, 6, 7\}$.

The structure of a MN encodes graphically a set of conditional independencies among the variables in the domain. These independencies are a valuable source of information in a number of fields 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], 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].

2 Motivation and Related Work

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]. 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. Independence-based algorithms rely on the fact that a graphical model implies that a set of independencies exist in the distribution of the domain, and therefore in the data set provided as input to the algorithm (under assumptions, see below); 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, that presents advantages in domains where independence tests are (a) uncertain or (b) expensive. The first case may occur in applications where data sets are small relative to the number of variables in the domain. Case (b) occurs in applications involving very large data sets (number

of data points) and/or 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 may be expensive, involving transfers of large amounts of data over a possibly slow network, so it is important to minimize the number of tests done.

Interesting work in the area of structure learning of undirected graphical models includes learning decomposable (also called chordal) MNs [Srebro and Karger, 2001] or more general (non-decomposable) MNs [Hofmann and Tresp, 1998], which is a score-based approach. An independence-based approach to MN structure learning is the GSIMN algorithm [Bromberg *et al.*, 2006], which uses Pearl's inference axioms [Pearl, 1988] to infer the result of certain independence tests without actually performing them. However, GSIMN has two disadvantages: (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]. In addition, our approach is more robust to errors in the test outcomes by making use of the probability of independence instead of making definite decisions (corresponding to probability 0 or 1). In this way an error in an independence test does not permanently cause the correct structure to be excluded but only lowers its posterior probability.

The rest of the paper is organized as follows: In the next section we present our notation, followed by a description of our approach in detail. Following that, we present experimental results and conclude with a summary of our approach.

3 Notation and Preliminaries

A Markov network of a domain $\mathcal{V}$ is an undirected model that can be used to represent the set of conditional independencies in the domain. The application domain is a set of random variables of size $n = |\mathcal{V}|$. In this work we use capital letters $A, B, \ldots$ to denote domain random variables and bold letters for sets of variables (e.g., $\mathbf{S}$). The space of all structures (given $\mathcal{V}$) is denoted by $\mathcal{X}$ and the space of all conditional independence (**CI**) tests by $\mathcal{Y}$. Conditional independence of A and B given $\mathbf{S}$ is denoted by $(A \perp\!\!\!\perp B \mid \mathbf{S})$. The set of conditional independencies implied by the structure of a MN are at least the ones that are implied by vertex separation i.e., $(A \perp\!\!\!\perp B \mid \mathbf{S})$ if A and B are separated in the MN graph after removing all nodes in $\mathbf{S}$ (and all edges adjacent to them). For example, in Fig. 1, $(0 \perp\!\!\!\perp 4 \mid \{2, 7\})$. If *exactly* those independencies hold in the actual probability distribution of the domain, we say that the domain and the graph are *faithful* to one another. 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.

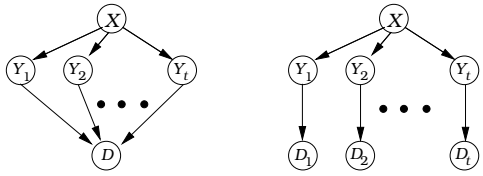

Figure 2: Generative model of domain. **Left:** Correct. **Right:** Assumed.

As described later in our main algorithm, we maintain populations of structures at each time step t; slightly abusing our notation we denote these populations by $\mathcal{X}_t$. We 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 $y_{1:t}$.

3.1 Generative Model over Structures and Tests

For an input data set $\mathcal{D}$, our approach uses the posterior probability over structures $\Pr(X \mid \mathcal{D}), X \in \mathcal{X}$ to learn the structure of the underlying model as explained in more detail in the next section. For that probability to be calculated in a principled way, we need a generative model that involves variables $X, Y_1, Y_2, \ldots, Y_t$ and $\mathcal{D}$ that reflects any independence constraints among these variables. Our only constraint here is the assumption that the tests are the *sufficient statistics* for the structure i.e., there is no information in the data set $\mathcal{D}$ beyond the value of the tests as far as structure is concerned. This stems from the fact that the structure X faithfully encodes the independencies in the domain. Note that this assumption is not particular to our approach, but is implicit in any independence-based approach. Our generative model only formalizes it and makes it explicit.

The generative model that encodes this constraint is shown in Fig. 2 (left), where X and $\mathcal{D}$ are d-separated by the tests $Y_1, \ldots, Y_t$. However, the posterior over structures $\Pr(X \mid \mathcal{D})$ cannot be computed from this model because, according to the model, $\Pr(X \mid \mathcal{D}) = \sum_{y_{1:t}} \Pr(X \mid Y_{1:t} = y_{1:t}, \mathcal{D}) \Pr(Y_{1:t} = y_{1:t} \mid \mathcal{D})$, which requires the computation of $\Pr(Y_{1:t} = y_{1:t} \mid \mathcal{D})$, currently an unsolved problem. We therefore assume the model shown in Fig. 2 (right), which contains multiple data sets $D_1, \ldots, \mathcal{D}_t$ (abbreviated $\mathcal{D}_{1:t}$). In this model, it can be shown that the tests Y_1 through Y_t are independent given data sets $\mathcal{D}_{1:t}$. This allows the model to be solved because now $\Pr(Y_{1:t} = y_{1:t} \mid \mathcal{D}) = \prod_{i=1}^{t} \Pr(Y_i = y_i \mid \mathcal{D}_i)$, where the factors $\Pr(Y_i = y_i \mid D_i)$ can be computed by known procedures such as the discrete version of the Bayesian test of [Margaritis, 2005]. In practice we do not have more than one data set, and therefore we use the same data set for all tests i.e., $\mathcal{D}_i = \mathcal{D}_j$, $i \neq j$. Thus, the model depicted on Fig. 2 (right) is used only as an approximation to overcome the lack of an exact solution described above. As we will show in the experiments section, this approximation works well in both artificial and real world data sets.

Under this modified model, the posterior probability over structures must now be computed given data sets $\mathcal{D}_{1:t}$, i.e., $\Pr(X \mid \mathcal{D}_{1:t})$, which we abbreviate as $\Pr_t(X)$. In our cal-

culations below we also use the conditional entropy $H(X \mid Y_{1:t}, \mathcal{D}_{1:t})$ of X given the set of tests $Y_{1:t}$ and the data sets $\mathcal{D}_{1:t}$, similarly abbreviated as $H_t(X \,|\, Y_{1:t})$.

Finally, the other parameters of the model are as follows: the prior $\Pr(X)$ is assumed uniform, and each test Y_i is completely determined given $X = x$ i.e., $\Pr(Y_i = \texttt{true} \mid X = x) \in \{0, 1\}$ (this is a direct consequence of the Faithfulness assumption).

4 Approach

To learn the MN structure of a domain from data, we employ a Bayesian approach, calculating the posterior probability $\Pr_t(x)$ of a structure $x \in \mathcal{X}$ given data sets $\mathcal{D}_{1:t}$. The problem of learning a structure in this framework can be summarized in the following two steps: (i) finding a sequence of tests $Y_{1:t}$ of minimum cost such that $H(X \mid Y_{1:t}, \mathcal{D}_{1:t}) = 0$, and (ii) finding the (unique) structure $x^\star$ such that $\Pr(X = x^\star \mid \mathcal{D}_{1:t}) = 1$. (This is always 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.)

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 $\binom{n}{2}\binom{n-2}{m}$ 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 sequences $Y_{1:t}$ to consider.

We address the first issue using a particle filtering approach (discussed in section 4.2). At each step t, we maintain a population of candidate MN structures $\mathcal{X}_t$ 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 second issue (choosing the next test to perform) is addressed using a greedy approach. At each step of our algorithm, we choose as the next test to perform the member of $\mathcal{Y}$ that minimizes the expected entropy, 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 explains our algorithm in detail.

4.1 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 randomly and uniformly selecting a number m of edges from 0 to $\binom{n}{2}$, and then randomly and uniformly selecting the m edges by picking the first m pairs in a random permutation of all possible pairs. This ensures that all edge-set sizes have equal probability of being represented in $\mathcal{X}_0$.

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 structure particles uniformly distributed among all edge-set sizes (see text).
2: $t \longleftarrow 0$
3: **loop**
4: $\quad Y_{t+1} \longleftarrow \arg\max_{(A,B)} \arg\max_{\mathbf{S}} score_t(A, B \mid \mathbf{S})$
5: $\quad Y_{1:t+1} \longleftarrow Y_{1:t} \cup \{Y_{t+1}\}$
6: $\quad p_T \longleftarrow \Pr(\mathcal{D}_{t+1} \mid Y_{t+1} = \texttt{t})$ /* *Perform test on data.* */
7: $\quad p_F \longleftarrow \Pr(\mathcal{D}_{t+1} \mid Y_{t+1} = \texttt{f})$ /* *Perform test on data.* */
8: $\quad$ Update $\Pr_{t+1}(X)$ from p_T and p_F using Eq. (5).
9: $\quad \mathcal{X}_{t+1} \longleftarrow PF(\mathcal{X}_t, M, \Pr_{t+1}(X), q(X^\star \mid X))$
10: $\quad$ **if** $H_{t+1}(X \mid Y_{1:t+1}) = 0$ **then**
11: $\quad\quad$ **return** unique structure x such that $\Pr_t(x) = 1$
12: $\quad t \longleftarrow t + 1$

At each time t during the main loop of the algorithm (lines 3–12), the test $Y_{t+1} = (A^\star, B^\star \mid \mathbf{S}^\star) \in \mathcal{Y}$ that optimizes a score function is selected (the score function is described below). Since for each pair of variables $(A, B) \in \mathcal{V} \times \mathcal{V}$ the space of possible conditioning sets is exponential (equaling the power set of $\mathcal{V} - \{A, B\}$), this optimization is performed by heuristic search; in particular, first-choice hill-climbing is used. During this procedure, the neighbors of a current point $\mathbf{S}$ are all possible additions to $\mathbf{S}$ of a non-member, all possible removals from $\mathbf{S}$ of a member, and all possible replacements of a member of $\mathbf{S}$ by a non-member. The score of test Y_{t+1} is defined as follows:

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

where the factor $W(Y)$ denotes the cost of Y, which we take to be proportional to the number of variables involved in the test. This factor is used to discourage expensive tests. $H_t(X \mid Y_{1:t}, Y_{t+1})$ is the entropy given the (not yet performed) test Y_{t+1}, and is equal to:

$$H_t(X \mid Y_{1:t}, Y_{t+1}) = H_t(X \mid Y_{1:t}) - IG_t(Y_{t+1}). \tag{2}$$

The derivation of Eq. (2) makes use of our generative model (Fig. 2) and is simple and omitted due to lack of space. The term $IG_t(Y_{t+1})$ denotes the information gain of candidate test Y_{t+1} given all information at time t, i.e., data sets $\mathcal{D}_{1:t}$, and is equal to:

$$IG_t(Y_{t+1}) = -\sum_{x \in \mathcal{X}} \Pr_t(x) \log \Pr_t(y^x_{t+1} \mid y^x_{1:t}) \tag{3}$$

where by $y^x_{1:t}$ we denote the value of tests $Y_{1:t}$ in structure x (each test is either `true` or `false`). The above expression follows from Bayes' rule and our generative model which implies that given a MN structure x, any test Y is completely determined using vertex separation (assuming Faithfulness). This implies that $\Pr_t(Y_{1:t} = y_{1:t} \mid x) = \delta(y_{1:t}, y^x_{1:t})$, where $\delta(a, b)$ is the Kronecker delta function that equals 1 iff a is equal to b. We say that structure x is *consistent* with assignment $y_{1:t}$ iff $y^x_{1:t} = y_{1:t}$. The quantity $\Pr_t(y^x_{t+1} \mid y^x_{1:t})$ of Eq. (3) can be calculated as

$$\Pr_t(y^x_{t+1} \mid y^x_{1:t}) = \frac{\sum_{x' \in \{y^x_{t+1}, y^x_{1:t}\}} \Pr_t(x')}{\sum_{x' \in \{y^x_{1:t}\}} \Pr_t(x')} \tag{4}$$

where by $\{y_{1:t}\}$ we denote the *equivalence class* of structures $x' \in \mathcal{X}$ that are consistent with assignment $y_{1:t}$. Using this

notation, $\{y^x_{1:t}\}$ represents the set of all structures that imply the same values for the CI tests $Y_{1:t}$ as structure x does. If x and x' are in the same class $\{y_{1:t}\}$, we say they are *equivalent* w.r.t. tests $Y_{1:t}$ and we denote this by $x \sim_t x'$.

Eq. (4) has an interesting and intuitive interpretation: at each time t, the posterior probability of a test Y being true (false), given some assignment $y_{1:t}$ of tests $Y_{1:t}$, equals the total posterior probability mass of the structures in the equivalence class $\{y_{1:t}\}$ that are consistent with $Y = \texttt{true}$ ($Y = \texttt{false}$), normalized by the posterior probability mass of the class $\{y_{1:t}\}$.

To compute the information gain in Eq. (2) we also need the posterior $\Pr_t(x) = \Pr(x \mid \mathcal{D}_{1:t})$ of a structure $x \in \mathcal{X}$ for Eq. (3). As explained in section 3.1, we use the Bayesian test described in [Margaritis, 2005]). This test calculates and compares the likelihoods of two competing multinomial models (with different numbers of parameters), namely $\Pr(\mathcal{D}_t \mid Y_t = y_t)$, $y_t \in \{\texttt{true}, \texttt{false}\}$. Since according to our generative model $\Pr(\mathcal{D}_i \mid y_{1:t}, x) = \Pr(\mathcal{D}_i \mid y_i)$ and $\Pr(y_{1:t} \mid x) = \delta(y^x_{1:t}, y_{1:t})$, by Bayes' law we have:

$$\Pr(x \mid \mathcal{D}_{1:t}) \propto \Pr(x) \Pr(\mathcal{D}_{1:t} \mid x) = \Pr(x) \prod_{i=1}^{t} \Pr(\mathcal{D}_i \mid y^x_i). \tag{5}$$

The constant of proportionality is independent of x and thus can be calculated by a sum over all structures in $\mathcal{X}$. As explained above, this, like all equations in this section that involve summations over the space of structures $\mathcal{X}$, is approximated by a summation over $\mathcal{X}_t$, the population of particles from the previous iteration of the algorithm.

This completes the description of test score function of Eq. (1). Returning to our description of Alg. 1, lines 6 and 7 calculate the data likelihoods of the optimal test Y_{t+1}, which are used to update the posterior over structures and obtain $\Pr_{t+1}(X)$ using Eq. (5). With this updated distribution, the new set of particles $\mathcal{X}_{t+1}$ is computed in line 9 using the particle filter algorithm, described in the next section.

The PFMN algorithm terminates when the entropy $H_t(X \mid Y_{1:t})$ (estimated over the population $\mathcal{X}_t$) becomes 0, returning the unique structure particle whose posterior probability equals 1.

4.2 Particle filter for structures

A particle filter [Andrieu *et al.*, 2003] is a sequential Markov-Chain Monte-Carlo (MCMC) method that uses a set of samples, called *particles*, to represent a probability distribution that may change after each observation in a sequence of observations. At each step, an observation is performed and a new set of particles is obtained from the set of particles at the previous step. The new set represent the new (posterior) distribution given the sequence of observations so far. One of the advantages of this sequential approach is the often drastic reduction of the cost of sampling from the new distribution, relative to alternative (non-sequential) sampling approaches. In our case, the domain is $\mathcal{X}$ and particles represent structures. Observations correspond to the evaluation of a single CI test on data.

The particle filter algorithm, shown in Algorithm 2, is used in line 9 of PFMN to transform population $\mathcal{X}_t$ to $\mathcal{X}_{t+1}$. The change in the probability distribution from $\Pr_t(X)$ to

Algorithm 2 Particle filter algorithm. $\mathcal{X}'' = PF(\mathcal{X}, M, f, q)$.

1: $\Pr_t(X) \longleftarrow f(X)$
2: **for all** particles $x \in \mathcal{X}$ **do**
3: $\quad w(x) \longleftarrow \frac{\Pr_t(x)}{\Pr_{t-1}(x)}$ */* Compute weights. */*
4: **for all** particles $x \in \mathcal{X}$ **do**
5: $\quad \widetilde{w}(x) \longleftarrow \frac{w(x)}{\sum_{x' \in \mathcal{X}} w(x')}$ */* Normalize weights. */*
6: */* Resample particles in $\mathcal{X}$ using $\widetilde{w}$ as the sampling probabilities. */*
7: $\mathcal{X}' \longleftarrow$ *resample*$(\mathcal{X}, \widetilde{w})$
8: */* Move each particle in $\mathcal{X}'$ M times using Metropolis-Hastings and distribution $\Pr_t(X)$. */*
9: $\mathcal{X}'' \longleftarrow \varnothing$
10: **for all** $x \in \mathcal{X}'$ **do**
11: $\quad \mathcal{X}'' \longleftarrow \mathcal{X}'' \cup$ *M-H*$(x, M, \Pr_t, q)$.
12: **return** $\mathcal{X}''$

Algorithm 3 Metropolis-Hastings algorithm. $x' = M - H(x, M, p, q)$.

1: $x^{(0)} \longleftarrow x$
2: **for** $i = 0$ to $M - 1$ **do**
3: $\quad u \sim \mathcal{U}_{[0,1]}$.
4: $\quad x^\star \sim q(x^\star \mid x^{(i)})$.
5: $\quad$ **if** $u < \mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p(x^\star)q(x^{(i)}|x^\star)}{p(x^i)q(x^\star|x^{(i)})}\right\}$ **then**
6: $\quad\quad x^{(i+1)} \longleftarrow x^\star$
7: $\quad$ **else**
8: $\quad\quad x^{(i+1)} \longleftarrow x^{(i)}$
9: $x' \longleftarrow x^{(M)}$
10: **return** x'

$\Pr_{t+1}(X)$ is reflected in a "weight." This weight is used as a probability in the resampling step (line 7) to bias selection (with replacement) among the particles in $\mathcal{X}_t$. In the final step, all particles are "moved" (lines 9–11) through M pairs of proposal/acceptance steps within the Metropolis-Hastings (**M-H**) algorithm, shown in Alg. 3 [Andrieu *et al.*, 2003] (in our experiments we use $M = 16$). This algorithm requires that $\Pr_t(X)$ and a *proposal* distribution $q(X^\star \mid X)$ be provided as parameters. $\Pr_t(X)$ has already been addressed above in Eq. (5). As in many particle filtering applications, the proposal distribution is a key factor for the success of PFMN, and is discussed in detail in the next section.

4.3 Proposal distribution for structures

As discussed previously, to use the Metropolis-Hastings algorithm one must provide a proposal distribution. Here, we use a mixture proposal $q = p_a q_a + (1 - p_a)q_b$, consisting of a global component q_a and a local component q_b. (We used $p_a = 0.05$ in our experiments). The components q_a and q_b are as follows:

- **Global proposal q_a (Random walk):** $q_a(x^\star \mid x)$ generates a sample $x^\star$ from x by iteratively inverting each edge of x with probability α. (In our experiments, we use $\alpha = 0.05$). 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 structures), we have that $q_a(x^\star \mid x) = \alpha^m(1-\alpha)^{m-h}$.
- **Local proposal q_b (Equivalence-class moves):** $q_b(x^\star \mid x)$ chooses to either (a) remove an edge from set $A^-(x)$

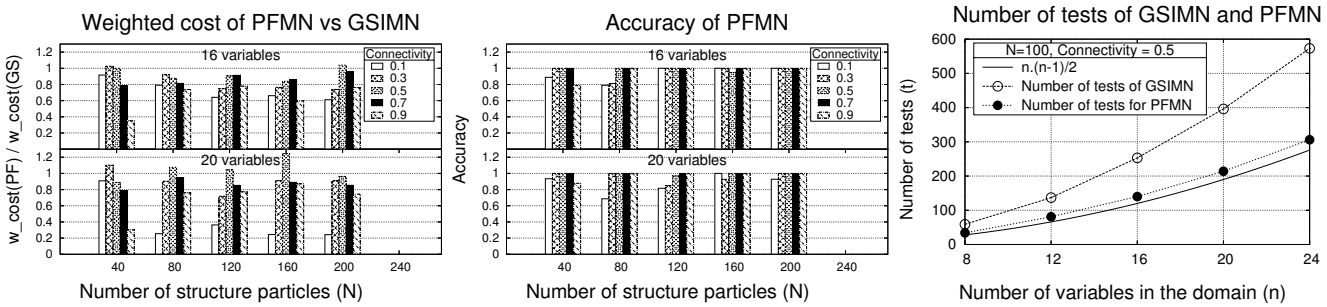

Figure 3: Performance comparison of PFMN vs. GSIMN for artificial domains. **Left:** Ratio of weighted cost for $n = 16, 20$ and $\tau = 0.1, 0.3, 0.5, 0.7, 0.9$. **Middle:** Accuracy for $n = 16, 20$ and $\tau = 0.1, 0.3, 0.5, 0.7, 0.9$. **Right:** Number of tests of conducted by PFMN and GSIMN compared to $\binom{n}{2}$.

(described below), (b) add an edge from set $A^+(x)$, or (c) remove an edge e and add e' where $(e, e') \in A^{\pm}(x)$. Each of (a), (b), or (c) is chosen with probability 1/3. Sets $A^+(x)$, $A^-(x)$, and $A^{\pm}(x)$ are defined as:

$$A^-(x) = \{e \in E(x) \mid x' = (\mathcal{V}, E(x) - \{e\}), x' \sim_t x\}$$
$$A^+(x) = \{e \in \bar{E}(x) \mid x' = (\mathcal{V}, E(x) \cup \{e\}), x' \sim_t x\}$$
$$A^{\pm}(x) = \{(e, e') \mid e \in E(x), e' \in E(x) \cup \{e\}, x' = (\mathcal{V}, (E(x) - \{e\}) \cup \{e'\}), x' \sim_t x\}$$

where $E(x)$ is the set of edges of structure x and $\bar{E}(x)$ is its complement. The set $A^+(x)$ ($A^-(x)$) is the set of all edges that are missing (existing) in x that if added to (removed from) x, produce a structure that will be equivalent to x i.e., in class $\{y^x_{1:t}\}$. The set $A^{\pm}(x)$ is the set of edge pairs (e, e') such that if e is removed and e' added to x the resulting structure will also be equivalent. Therefore the result of any move under this proposal is a structure that is guaranteed to be equivalent to x, with Hamming distance from x of at most 1.

The local proposal was designed to maximize acceptance of proposed moves (i.e., line 5 of the M-H algorithm evaluating to `true`) by proposing moves to independence-equivalent structures, which have provably the same posterior probability. A well-designed proposal $q(X^\star \mid X)$ i.e., one that ensures convergence to the posterior distribution $\Pr_t(X)$, must also 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 Experiments

We experimentally compare PFMN with the GSIMN algorithm of [Bromberg *et al.*, 2006] on both artificial and real-world data sets. GSIMN is a state-of-the-art exact independence-based algorithm that uses Pearl's axioms of CI tests to infer certain tests without actually conducting them. We report: (a) *weighted number of tests*, and (b) *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 [Bromberg *et al.*, 2006] for details. To obtain the quality of the recovered network, we measure accuracy as the fraction of *unseen* CI tests that are correct i.e., we compare the result of each test on the resulting structure (using vertex separation) with the true value of the test (calculated using vertex separation in the true model for artificial domains or statistical tests on the data for real-world domains). The purpose of this procedure is to measure how well the output network generalizes to unseen tests.

5.1 Artificial Experiments

We first evaluated our algorithm in artificial domains in which the structure of the underlying model, called *true network*, is known. This allowed (i) a systematic study of its behavior under varying conditions of domain sizes n and amount of dependencies (reflect in the number of edges m in the true network), and, (ii) a better evaluation of quality of the output networks because the true model is known. True networks were generated randomly by selecting the first $m = \tau\binom{n}{2}$ pairs in a random permutation of all possible edges, τ being a *connectivity* parameter.

Fig. 3 shows the results of experiments for which CI tests were conducted directly on the true network through vertex-separation, i.e., outcomes of tests are always correct. Fig. 3 (left) shows the ratio of the weighted cost of CI tests conducted by PFMN compared to GSIMN for two domain sizes $n = 16$ (above) and $n = 20$ (below), a number of connectivities ($\tau = 0.1, 0.3, 0.5, 0.7$, and 0.9), and an increasing number of structure particles N. We can see that weighted cost of PFMN is lower than that of GSIMN for small ($\tau = 0.1, 0.3$) and large ($\tau = 0.7, 0.9$) connectivities, reaching savings of up to 78% for $n = 20$ and $\tau = 0.1$. The case of $\tau = 0.5$ is inconclusive—PFMN does better in some, but not all cases. This can be explained by the fact that the number of structures with connectivity τ grows exponentially as τ approaches 0.5 (from either side), which clearly makes the search task in PFMN more difficult. Fig. 3 (middle) shows that even though PFMN is an approximate algorithm, its accuracy for large enough number of particles N is exactly 1 in all but a few cases (e.g., $N = 160, n = 16, \tau = 0.5$). GSIMN is an exact algorithm, i.e., its accuracy is always 1 when independence tests are correct, which is the case for the above experiments. Its accuracy is thus omitted from the figures. Fig. 3 (right) shows that PFMN does much fewer (non-weighted) test than GSIMN, and very close to the optimal number of tests which

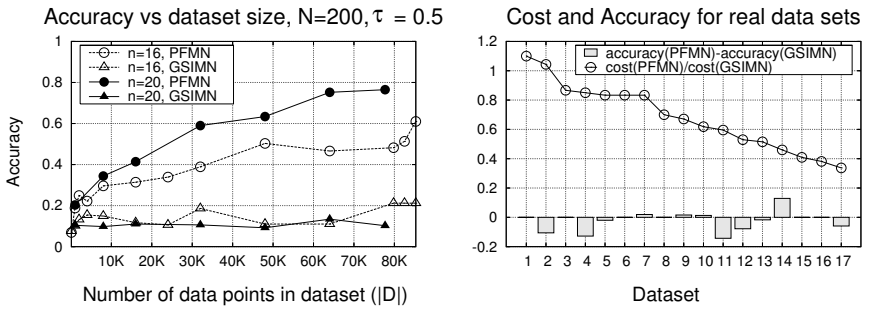

Figure 4: Performance comparison of PFMN and GSIMN on data. **Left:** Accuracy of PFMN and GSIMN vs. data set size for sampled data. **Right:** Ratio of weighted cost of PFMN vs. GSIMN and difference between accuracy of PFMN and GSIMN on real data sets. The numbers on the x-axis correspond to data set indices in Table 1.

Table 1: Comparison of PFMN vs. GSIMN for real-world data.

Data set				w_cost		Accuracy	
#	name	n	$\|\mathcal{D}\|$	GSIMN	PFMN	GSIMN	PFMN
1	haberman	5	305	**20**	22	0.971	0.971
2	adult	10	32561	**138**	144	**0.785**	0.679
3	hayes-roth	6	132	30	**26**	0.831	0.831
4	balance-scale	5	625	20	**17**	**0.871**	0.743
5	nursery	9	12960	72	**60**	**0.878**	0.858
6	monks-1	7	556	42	**35**	0.912	0.912
7	car	7	1728	42	**35**	0.826	**0.845**
8	baloons	5	20	20	**14**	0.857	0.857
9	crx	16	653	240	**161**	0.898	**0.914**
10	hepatitis	20	80	380	**235**	0.876	**0.889**
11	cmc	10	1473	126	**75**	**0.893**	0.750
12	tic-tac-toe	10	958	102	**54**	**0.836**	0.758
13	bridges	12	70	132	**68**	**0.844**	0.827
14	alarm	37	20001	1335	**614**	0.797	**0.926**
15	imports-85	25	193	633	**259**	0.688	0.688
16	flag	29	194	839	**320**	0.878	0.878
17	dermatology	35	358	1211	**408**	**0.813**	0.754

is $\log 2^{\binom{n}{2}} = \binom{n}{2}$ assuming a uniform distribution over structures, since the size of the space of structures is $2^{\binom{n}{2}}$. The figure shows that PFMN is very close to this optimal for a range of domain sizes n.

We also compared the robustness of PFMN and GSIMN to uncertainty in the outcome of tests. This uncertainty is expected to occur in small data sets. Data sets of different sizes were sampled from true networks of $n = 20$ and $n = 16$ variables both of connectivity $\tau = 0.5$. Fig. 4 (left) shows the accuracy of PFMN and GSIMN vs. the size of the data set ($|D|$). Accuracy of PFMN clearly outperforms that of GSIMN, with the gap between their accuracies growing steadily with the data set size for both $n = 16$ and $n = 20$.

5.2 Real-World Experiments

We also conducted experiments on a substantial number of data sets obtained from the UCI ML archive [D.J. Newman and Merz, 1998]. As above, we compare the weighted cost and accuracy of PFMN vs. GSIMN. Fig. 4 (right) shows the ratio of the weighted cost of PFMN vs. GSIMN (i.e., a number smaller than 1 shows improvement of PFMN vs. GSIMN) and the difference in accuracies of PFMN and GSIMN (i.e., a positive histogram bar shows an improvement of PFMN vs. GSIMN). In these experiments, PFMN used $N = 200$. The numbers in the x-axis are indices to the data sets shown in table 1. For 15 out of 17 data sets, PFMN required lower weighted cost, reaching ratios as low as 0.38. Moreover, for 11 out 17 data sets PFMN achieves this reduction in cost with little or no reduction in accuracy compared to GSIMN, the rest exhibit only a modest reduction (less than 15%).

6 Summary and Conclusion

In this paper we presented PFMN, an independence-based algorithm for learning the structure of a Markov network from data. We presented an analysis of the domain of structures and independencies and a number of interesting relations using an explicit generative model. We also showed experimentally that, compared to existing independence-based algorithms, PFMN is more resistant to errors in the test conducted and executes fewer tests in many cases. This helps in domains where data are scarce and tests uncertain or data are abundant and/or distributed over a potentially slow network.

References

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

[Besag *et al.*, 1991] 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.

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

[D.J. Newman and Merz, 1998] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases. 1998.

[Geman and Geman, 1984] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian relation of images. *PAMI*, 6:721–741, 1984.

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

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

[Margaritis, 2005] D. Margaritis. Distribution-free learning of Bayesian network structure in continuous domains. In *AAAI*. AAAI Press, 2005.

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

[Shekhar *et al.*, 2004] S. Shekhar, P. Zhang, Y. Huang, and R. Vatsavai. Trends in spatial data mining. chapter 19, pages 357–379. AAAI Press / MIT Press, 2004.

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

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

[Tong and Koller, 2001] S. Tong and D. Koller. Active learning for structure in Bayesian networks. In *IJCAI*, 2001.