Marginal Pseudo-Likelihood Learning of Markov Network Structures

Johan Pensar JOHAN.PENSAR@ABO.FI
Henrik Nyman HENRIK.NYMAN@ABO.FI
Department of Mathematics and statistics
Åbo Akademi University
20500 Turku, Finland

Juha Niiranen JUHA.NIIRANEN@HELSINKI.FI
Jukka Corander JUKKA.CORANDER@HELSINKI.FI
Department of Mathematics and statistics
University of Helsinki
00014 Helsinki, Finland

Editor:

arXiv:1401.4988v2 [stat.ML] 11 Nov 2014

Abstract

Undirected graphical models known as Markov networks are popular for a wide variety of applications ranging from statistical physics to computational biology. Traditionally, learning of the network structure has been done under the assumption of chordality which ensures that efficient scoring methods can be used. In general, non-chordal graphs have intractable normalizing constants which renders the calculation of Bayesian and other scores difficult beyond very small-scale systems. Recently, there has been a surge of interest towards the use of regularized pseudo-likelihood methods for structural learning of large-scale Markov network models, as such an approach avoids the assumption of chordality. The currently available methods typically necessitate the use of a tuning parameter to adapt the level of regularization for a particular dataset, which can be optimized for example by cross-validation. Here we introduce a Bayesian version of pseudo-likelihood scoring of Markov networks, which enables an automatic regularization through marginalization over the nuisance parameters in the model. We prove consistency of the resulting MPL estimator for the network structure via comparison with the pseudo information criterion. Identification of the MPL-optimal network on a prescanned graph space is considered with both greedy hill climbing and exact pseudo-Boolean optimization algorithms. We find that for reasonable sample sizes the hill climbing approach most often identifies networks that are at a negligible distance from the restricted global optimum. Using synthetic and existing benchmark networks, the marginal pseudo-likelihood method is shown to generally perform favorably against recent popular inference methods for Markov networks.

Keywords: Bayesian inference, Markov networks, structure learning, undirected graph, pseudo-likelihood, regularization

1. Introduction

Markov networks represent a ubiquitous modeling framework for multivariate systems, with applications ranging from statistical physics to computational biology and sociology (see

Lauritzen, 1996; Koller and Friedman, 2009). However, statistical inference for such models is in general challenging, both regarding estimation of parameters and learning structure of the network. Under the assumption of chordality it is possible to use a closed-form factorization of a distribution with respect to a Markov network, however, in non-chordal cases the normalizing factor (or partition function) of these distributions is intractable beyond toy-sized systems. Since the chordality assumption is restrictive and may seriously bias learning of the dependencies among variables, considerable interest has been targeted towards making also non-chordal networks tractable for applications. A revival of interest has in particular arisen from the need to consider high-dimensional models in a 'large p, small n' setting (Lee et al., 2006; Höfling and Tibshirani, 2009; Ravikumar et al., 2010; Aurell and Ekeberg, 2012; Ekeberg et al., 2013).

In physics, Markov network models have traditionally been fitted using the mean-field approximation, which has only recently started to become superceded by more elaborate approaches, such as the pseudo-likelihood method (Aurell and Ekeberg, 2012; Ekeberg et al., 2013). The pseudo-likelihood approach was originally motivated by the difficulties of maximizing the likelihood function for lattice models (Besag, 1972) and it simplifies the model fitting by a factorization of the likelihood over local neighborhoods of the random variables involved in the modeled system.

High-dimensional Markov networks usually necessitate some form of regularization to make the pseudo-likelihood estimation problem feasible to solve. Some of the currently available methods necessitate the use of a tuning parameter to adapt the level of regularization for a particular dataset. The value of the tuning parameter can then be optimized for example by cross-validation. Here we introduce a Bayesian version of the pseudo-likelihood approach to learn the structure of a Markov network without assuming chordality. Our method enables an automatic regularization of the resulting model complexity through marginalization over the nuisance parameters in the model.

The structure of the remaining article is as follows. In the next section the basic properties of Markov networks are reviewed and the structure learning problem is formulated in Section 3. In Section 4, we introduce the marginal pseudo-likelihood (MPL) score and prove consistency of the corresponding structure estimator. Algorithms for optimizing the MPL score for a given dataset are derived in Section 5 and the penultimate section demonstrates the favorable performance of our method against other popular recent alternatives. The last section provides some additional remarks and conclusions.

2. Markov networks

We consider a set of d discrete random variables $X = \{X_1, \ldots, X_d\}$ where each variable X_j takes values from a finite set of outcomes $\mathcal{X}_j$. A Markov network over X is a undirected probabilistic graphical model that compactly represents a joint distribution over the variables. The dependence structure over the d variables is specified by an undirected graph $G = (V, E)$ where the nodes $V = \{1, \ldots, d\}$ correspond to the indices of the variables X and the edge set $E \subseteq \{V \times V\}$ represents dependencies among the variables. We will use the terms node and variable interchangeably throughout this article. The complete set of undirected graphs is denoted by $\mathcal{G}$.

A node i is a neighbor of j (and vice versa) if $\{i,j\} \in E$ and the set of all neighbors of j is called its Markov blanket, which is denoted by $mb(j) = \{i \in V : \{i,j\} \in E\}$. A clique in a graph is a subset of nodes, $C \subseteq V$, for which every pair of nodes are connected by an edge, that is $\{i,j\} \in E$ if $i,j \in C$. A clique is considered maximal if it cannot be extended by including an additional node without violating the clique criterion. The set of maximal cliques associated with a graph is denoted by $\mathcal{C}(G)$. The variables corresponding to a subset of nodes, $S \subseteq V$, are denoted by $X_S = \{X_j\}_{j \in S}$ and the corresponding joint outcome space is specified by the Cartesian product $\mathcal{X}_S = \times_{j \in S} \mathcal{X}_j$. The cardinality of an outcome space is denoted by $|\mathcal{X}_S|$. We use a lowercase letter x_S to denote that the variables have been assigned a specific joint outcome in $\mathcal{X}_S$. A dataset $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ refers to a collection of n i.i.d. complete joint observations $\mathbf{x}_k = (x_{k,1}, \ldots, x_{k,d})$ over the d variables, that is $x_{k,j} \in \mathcal{X}_j$ for all k and j.

In addition to the graph, to fully specify a Markov network one must also define a probability distribution that satisfies the restrictions imposed by the graph G. We restrict the models to positive and faithful distributions unless otherwise mentioned. A distribution is said to be faithful to G if it does not satisfy any additional independencies that are not conveyed by the graph. In this case G can be considered a true representation in the sense that no artificial dependencies are introduced. We use θ_G to denote the set of parameters describing a distribution of a model with graph G. The parameter space Θ_G contains all possible instantiations of θ_G corresponding to a distribution satisfying G. Finally, we use $p(x_A \mid x_B)$ as an abbreviated notation for the conditional probability $p(X_A = x_A \mid X_B = x_B)$, while $p(X_A \mid X_B)$ represents the corresponding family of conditional distributions.

The concept of graphical models is based on the assumption of modularity manifested in the factorization of the joint distribution. In particular, the (positive) joint distribution of a Markov network can be factorized over the maximal cliques in the graph according to

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \phi(x_C) \tag{1}$$

where $\phi(x_C) : \mathcal{X}_C \to \mathbb{R}_+$ is a clique factor (or potential) and $Z = \sum_{x \in \mathcal{X}} \prod_{C \in \mathcal{C}(G)} \phi(x_C)$ is a normalizing constant known as the partition function. Markov networks are often also parameterized in terms of a log-linear model in which each clique factor is replaced by an exponentiated weighted sum of features according to

$$p(x) = \frac{1}{Z} \exp\left(\sum_{f_K \in \mathcal{F}} w_K f_K(x_K) \right) \tag{2}$$

where $\mathcal{F} = \{f_K\}$ is the set of feature functions and $\mathcal{W} = \{w_K\}$ is the corresponding set of weights. A feature function $f_K : \mathcal{X}_K \to \mathbb{R}$ maps each value $x_K \in \mathcal{X}_K$ for some $K \subseteq V$ to a numerical value, typically it is in the form of an indicator function that equals 1 if the value matches a specific feature and 0 otherwise. Every Markov network can be encoded as a log-linear model by defining a feature as an indicator function for every assignment of X_C for each $C \in \mathcal{C}(G)$. In this case, the weights in (2) correspond to the natural logarithm of the clique factors in (1). Conversely, a log-linear model over X implicitly induces the graph of a Markov network by imposing an edge $\{i,j\}$ for every pair of variables appearing in the same domain of some feature function $f_K(x_K)$, that is $\{i,j\} \in E$ if $\{i,j\} \subseteq K$.

The absence of edges in the graph $G = (V, E)$ of a Markov network encodes statements of conditional independence. The variables X_A are conditionally independent of the variables X_B given the variables X_S if $p(X_A \mid X_B, X_S) = p(X_A \mid X_S)$ holds. We denote this by

$$X_A \perp X_B \mid X_S.$$

The dependence structure of a Markov network can be characterized by the following Markov properties:

1. Pairwise Markov property: $X_i \perp X_j \mid X_{V \setminus \{i,j\}}$ for all $\{i, j\} \notin E$.
2. Local Markov property: $X_i \perp X_{V \setminus \{mb(i) \cup i\}} \mid X_{mb(i)}$ for all $i \in V$.
3. Global Markov property: $X_A \perp X_B \mid X_S$ for all disjoint subsets (A, B, S) of V such that S separates A from B.

Although the strength of the above properties differ in general, they are proven to be equivalent under the current assumption of positivity of the joint distribution (Lauritzen, 1996). While the last property is sufficient in the sense that it captures the entire set of independencies induced by a network, the first two properties are also useful since they allow one to focus on smaller sets of independencies. In particular, our MPL approach for structure learning is based on the local Markov property.

3. Structure learning

There are two main tasks associated with fitting graphical models to data; parameter estimation and structure learning. In this work, we focus entirely on the latter. By structure learning, we refer to the process of deducing the dependence structure from a set of data assumed to be generated from an unknown Markov network. The structure learning problem can be considered a model class learning problem in the sense that each specific structure alone represents a class of models. In many applications, the structure is a goal in itself in the sense that one wants merely to gain a qualitative insight into the dependence structure of an underlying process. However, given a known structure, the problem of model parameter estimation is simplified. Hence, if the distribution needs also to be explicitly estimated, this can be achieved by using any of the several existing methods conditional on the fixed structure learned by our approach.

3.1 Hypothesis space

When learning the structure of a Markov network the considered space of model classes, or hypothesis space, can be formulated in terms of different degrees of granularity (Koller and Friedman, 2009). The most fine-grained structure learning methods aim at recovering distinct features in the log-linear parameterization (2), this approach is commonly referred to as feature selection (Pietra et al., 1997; Lee et al., 2006; Höfling and Tibshirani, 2009; Ravikumar et al., 2010; Lowd and Davis, 2014). The advantage of a very detailed structure is that it enables the model to better emulate the properties of a distribution without imposing redundant parameters. One possible drawback of this formulation is the risk of overfitting the structure through long specialized features. Since every pair of variables in a

feature results in an edge, such parameterizations can obscure the connection to the graph structure in the sense that sparsity in the number of features does not in general correspond to sparsity in the number of edges in the graph. In contrast to the very specific feature selection problem, the model space of our approach is formulated directly in terms of the graph structure alone and the complexity of a model is defined by the size of the maximal cliques in the network.

Although dense graphs are not necessarily unfavorable, there are several situations where a sparse graph is preferred. In particular, when the ultimate goal is knowledge discovery, a dense graph may in the worst case hide the primary layer of the dependency pattern. Another important aspect is the feasibility of performing probabilistic inference in the model. One of the main inference tasks for graphical models is the process of computing the posterior probability of a list of query variables given some observed variables. Inference methods designed for this purpose often exploit the sparsity of the graph structure and dense graphs inevitably hamper the efficiency of such algorithms.

3.2 Different approaches

Structure learning methods can roughly be divided into two categories; constraint-based and score-based methods. Constraint-based approaches aim at inferring the structure through a series of independence tests based on the Markov properties, (Spirtes et al., 2000; Tsamardinos et al., 2003; Bromberg et al., 2009; Anandkumar et al., 2012). This approach is appealing in the sense that the independently performed tests can be combined into a structure through a divide-and-conquer approach. Under the assumptions that the distribution is faithful to graph structure and that the tests are correct, the true structure can be reconstructed. However, constraint-based approaches can be quite sensitive to failures in individual tests in the sense that a wrong answer from an independence test can mislead the network construction procedure (Koller and Friedman, 2009). In practice, rather large sample sizes may be required for the independence tests to yield correct answers.

The score-based approach formulates structure learning as an optimization problem. One defines an objective or score function according to which the plausibility of each candidate in the model space can be evaluated. Since score functions consider the whole structure at once, they can be less sensitive to individual failures. The disadvantage of the score-based approach is that it usually requires use of an optimization algorithm. This poses an obvious problem since the search space for d nodes consists of $2^{\binom{d}{2}}$ distinct undirected graphs. Finding the global optimum in such enormous combinatorial spaces becomes intractable already for moderate-sized models. For this reason, a selection of heuristic search algorithms have been developed for the sole purpose of finding high-scoring networks and many of them have been shown to work well in practice.

The most commonly used objective function is the likelihood of the data given a graph,

$$l(\theta_G; \mathbf{x}) = p(\mathbf{x} \mid \theta_G) = \prod_{k=1}^{n} p(\mathbf{x}_k \mid \theta_G),$$

or, in practice, the corresponding log-likelihood function,

$$\ell(\theta_G; \mathbf{x}) = \log l(\theta_G; \mathbf{x}).$$

By maximizing the (log-)likelihood, the model fit to the data is maximized. Although there exists no analytical solution for non-chordal Markov networks, the concavity of the likelihood function enables it to be maximized by numerical optimization for moderate-sized models. The maximum likelihood alone is not an appropriate objective function since it obtains its maximum value under the complete graph due to noise in the data. One option is to constrain the expressiveness of the graphs in the model space, for example by only considering tree structures (Chow and Liu, 1968). A problem with such a constraint is that it may easily end up limiting the model space to networks not suitable for modeling the data. A more popular approach is to regulate the fit by adding a sparsity-promoting penalty function to the log-likelihood (Akaike, 1974; Schwarz, 1978; Lee et al., 2006).

In contrast to the above methods where the complexity of a model is penalized explicitly, the Bayesian framework provides an alternative by implicitly preventing overfitting. In the Bayesian approach a graph is scored by its posterior probability given the data,

$$p(G \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid G) \cdot p(G)}{p(\mathbf{x})}. \tag{3}$$

In practice it suffices to consider the unnormalized posterior probability

$$p(G, \mathbf{x}) = p(\mathbf{x} \mid G) \cdot p(G), \tag{4}$$

since $p(\mathbf{x})$ is a normalizing constant that can be ignored when comparing graphs. The key factor of (4) is $p(\mathbf{x} \mid G)$ which is the marginal likelihood (ML) of the data given the network structure (also called the evidence). To evaluate the ML, one must integrate the likelihood function over all parameter values satisfying the restrictions imposed by the graph according to

$$p(\mathbf{x} \mid G) = \int_{\Theta_G} l(\theta_G; \mathbf{x}) \cdot f(\theta_G) d\theta_G, \tag{5}$$

where $f(\theta_G)$ is a prior distribution that assigns a weight to each $\theta_G \in \Theta_G$. Since the ML accounts for the parameter uncertainty through the prior, it implicitly regulates the fit to the data against the complexity of the network.

A drawback of the ML is that it is extremely hard to evaluate for non-chordal Markov networks. For this reason various penalized maximum likelihood objectives have naturally been preferred. In particular, Schwarz (1978) introduced the Bayesian information criterion (BIC) as an asymptotic approximation of the ML. Still, due to the partition function, even maximum likelihood based techniques become intractable for larger models and require use of approximate inference. Therefore, in the next section we derive an alternative Bayesian-type score applicable also to very large systems.

Given a scoring function, it is still necessary to specify a search algorithm to find high-scoring networks since the discrete search space is in general too large for an exhaustive evaluation. To avoid the discrete nature of the model space, Lee et al. (2006) introduced an L_1-based penalty to reformulate the structure learning problem as a convex optimization problem over the continuous parameter space. This is an elegant technique that has been further developed (Höfling and Tibshirani, 2009; Ravikumar et al., 2010) for the special class of binary pairwise Markov networks for which the method is especially well-suited. Each edge in such a network is associated with a single parameter and forcing an edge

parameter to zero is equivalent to removing the corresponding edge from the network. In fact, in this case the problem formulation of feature selection and graph structure discovery are equivalent due to the one-to-one correspondence between edges and features. For more general networks, sparsity must be enforced to groups of parameters in order to achieve sparsity in the number of edges in the resulting graph (Schmidt and Murphy, 2010). For the direct approach of Lee et al. (2006) the issue of maximizing the likelihood function still remains.

The main difference between constraint- and score-based methods is the level at which they approach the problem (Koller and Friedman, 2009). Score-based methods works on a global level by considering the whole structure at once. This makes them less sensitive to individual failures but has a negative effect on their scalability. The local approach of constraint-based methods allows them to scale up well but it makes them more sensitive to failures in the individual tests. Although the MPL as such would fall into the score-based category, under the optimization strategy introduced in Section 5, our MPL method is rather a hybrid by which we aim to achieve scalability as well as reliable performance.

4. Marginal pseudo-likelihood

In order to avoid problems associated with the evaluation of the true likelihood function, one can preferably use alternative objectives that possess favorable properties from a computational perspective. In this work we consider the commonly used pseudo-likelihood, originally introduced by Besag (1972), from a Bayesian perspective.

4.1 Derivation

The pseudo-likelihood function approximates the likelihood function by a factorization into conditional likelihood functions according to

$$pl(\theta; \mathbf{x}) = \prod_{k=1}^{n} \prod_{j=1}^{d} p(x_{k,j} \mid x_{k,V\setminus j}, \theta).$$

For a fixed graph structure G, the local Markov property implies that a variable in a Markov network is independent of the remaining variables given its Markov blanket such that

$$p(X_j \mid X_{V\setminus j}, G) = p(X_j \mid X_{mb(j)}, G)$$

must hold. Consequently, the pseudo-likelihood for a fixed graph is given by

$$pl(\theta_G; \mathbf{x}) = \prod_{k=1}^{n} \prod_{j=1}^{d} p(x_{k,j} \mid x_{k,mb(j)}, \theta_G). \tag{6}$$

In terms of the log-linear parameterization (2), the pseudo-likelihood approximation offers huge computational savings compared to the true likelihood since the global normalizing constant in the likelihood function is replaced by d local normalizing constants. By replacing the likelihood with the pseudo-likelihood, methods originally based on the maximum likelihood have been extended to work on larger systems. For example, the pseudo-likelihood

approximation of Höfling and Tibshirani (2009) and the closely related method by Ravikumar et al. (2010) highlight how the original idea of Lee et al. (2006) can be extended to higher dimensions. Ji and Seymour (1996) and Csiszár and Talata (2006) both derived a pseudo-likelihood version of the Bayesian information criterion by Schwarz (1978). An encouraging aspect is that several pseudo-likelihood approaches have been shown to enjoy consistency under the assumption that the data is generated by a distribution in the model class (Ji and Seymour, 1996; Csiszár and Talata, 2006; Ravikumar et al., 2010).

From a Bayesian perspective, the structural form of (6) offers an interesting possibility. In fact, under certain assumptions it enables an analytical evaluation of the integral

$$\hat{p}(\mathbf{x} \mid G) = \int_{\Theta_G} pl(\theta_G; \mathbf{x}) \cdot f(\theta_G) d\theta_G \tag{7}$$

which is here referred to as the marginal pseudo-likelihood (MPL). We parameterize the conditional probabilities associated with the pseudo-likelihood function of a graph by

$$\theta_{ijl} = p(X_j = x_j^{(i)} \mid X_{mb(j)} = x_{mb(j)}^{(l)}) \text{ where } \theta_{ijl} > 0 \text{ and } \sum_{i=1}^{r_j} \theta_{ijl} = 1. \tag{8}$$

The indices $i = 1, \ldots, r_j$ and $l = 1, \ldots, q_j$, where $r_j = |\mathcal{X}_j|$ and $q_j = |\mathcal{X}_{mb(j)}| = \prod_{i \in mb(j)} r_i$, represent the configurations of the variable and its respective Markov blanket. The above set of graph-specific parameters is by no means a compact representation of a Markov network, in fact, it is a quite crude over-parameterization. Rather than actual model parameters, they should be considered temporary nuisance parameters, used solely for computational convenience, in solving the structure learning problem. Similarly as above, we denote the counts of the corresponding configurations in $\mathbf{x}$ by

$$n_{ijl} = \sum_{k=1}^{n} \mathbf{I}\left[(x_{k,j}, x_{k,mb(j)}) = (x_j^{(i)}, x_{mb(j)}^{(l)})\right] \text{ and } n_{jl} = \sum_{i=1}^{r_j} n_{ijl}.$$

The pseudo-likelihood function can now be expressed in terms of our above notation by

$$pl(\theta_G; \mathbf{x}) = \prod_{j=1}^{d} \prod_{l=1}^{q_j} \prod_{i=1}^{r_j} \theta_{ijl}^{n_{ijl}}. \tag{9}$$

Under the current parameterization it is easy to make out certain structural similarities between the above pseudo-likelihood function and the likelihood function of a Bayesian network under a standard conditional parameterization (see e.g. Koller and Friedman, 2009). In a Bayesian network the l-index would be associated with configurations of parent sets instead of configurations of Markov blankets. The parent sets must be such that they satisfy the acyclicity constraint imposed by a DAG whereas the Markov blankets must be mutually consistent. Under certain assumptions listed by Heckerman et al. (1995) the ML of a Bayesian network has a nice analytical expression that factorizes variable-wise making it attractive for the task of structure learning. Using a corresponding set of assumptions we would like to achieve something similar for the ML. We consider the parameters defined in (8) in terms of the sets

$$\theta_{jl} = \cup_{i=1}^{r_j} \{\theta_{ijl}\},\ \theta_j = \cup_{l=1}^{q_j} \{\theta_{jl}\}, \text{ and } \theta_G = \cup_{j=1}^{d} \{\theta_j\}.$$

One of the fundamental assumptions behind the ML for Bayesian networks is an assumption regarding global and local parameter independence (Assumption 2, Heckerman et al., 1995). This assumption ultimately justifies a factorization of the parameter prior. We need to factorize the parameter prior in (7) in a corresponding fashion according to

$$f(\theta_G) = \prod_{j=1}^{d} f(\theta_j) = \prod_{j=1}^{d} \prod_{l=1}^{q_j} f(\theta_{jl}),$$

implying that $\theta_j \perp \theta_{j'}$ for $j \neq j'$ (global parameter independence) and $\theta_{jl} \perp \theta_{jl'}$ for $l \neq l'$ (local parameter independence). Whereas parameter independence can be a quite reasonable assumption in a Bayesian network parameterization, in our case it directly violates the properties of a Markov network. The conditional distributions, represented by our parameters, are connected to each other in the sense that they must satisfy certain algebraic relations for them to be consistent with a Markov network. We do not elaborate on these relations but we note that they directly translate to restrictions between the corresponding parameter sets. At this point, the parameter independence assumption is mainly justified by the induced computational savings. In Section 4.4 we discuss the implications of the assumption more in detail from another perspective. Another fundamental assumption, necessary for our derivation, is to restrict each parameter set θ_{jl} to follow a Dirichlet distribution

$$\theta_{jl} \sim \text{Dirichlet}(\alpha_{1jl}, \ldots, \alpha_{r_j jl}),$$

where $\alpha_{1jl}, \ldots, \alpha_{r_j jl}$ are hyperparameters for which we denote $\alpha_{jl} = \sum_{i=1}^{r_j} \alpha_{ijl}$.

Under the established assumptions, the integral in (7) can be reordered into a product of local integrals. Since the Dirichlet distribution is a conjugate prior of the multinomial distribution, each local integral is easily solved using standard Bayesian calculations:

$$\begin{aligned}
\hat{p}(\mathbf{x} \mid G) &= \int_{\Theta_G} pl(\theta_G; \mathbf{x}) \cdot f(\theta_G) d\theta_G \\
&= \prod_{j=1}^{d} \prod_{l=1}^{q_j} \int_{\Theta_{jl}} \prod_{i=1}^{r_j} \theta_{ijl}^{n_{ijl}} \cdot f(\theta_{jl}) d\theta_{jl} \\
&= \prod_{j=1}^{d} \prod_{l=1}^{q_j} \frac{\Gamma(\alpha_{jl})}{\Gamma(n_{jl} + \alpha_{jl})} \prod_{i=1}^{r_j} \frac{\Gamma(n_{ijl} + \alpha_{ijl})}{\Gamma(\alpha_{ijl})}
\end{aligned}$$

In practice, the logarithm of the formula is used since it is computationally more manageable.

To evaluate the above expression it is necessary to define the hyperparameters. We want to specify a symmetric prior since we assume that there is no prior knowledge favoring one parameter in $\{\theta_{1jl}, \ldots, \theta_{r_j jl}\}$ over any of the others. We achieve this by modifying a prior originally defined for Bayesian networks by Buntine (1991) such that the hyperparameters are determined according to

$$\alpha_{ijl} = \frac{N}{|\mathcal{X}_j| \cdot |\mathcal{X}_{mb(j)}|} = \frac{N}{r_j \cdot q_j},$$

where N is the equivalent sample size adjusting the strength of the prior.

4.2 Properties

The MPL possesses several advantageous properties. The parameter prior offers a natural regularization that prevents overfitting. Methods that explicitly penalize the degree of regularization are sensitive to the choice of some tuning parameter, which usually has to be determined empirically. In contrast, the MPL requires specification of the hyperparameters in the Dirichlet distribution. In our formulation this boils down to setting a value on the equivalent sample size N. Silander et al. (2007) show that the maximum a posteriori (MAP) Bayesian network structure optimization problem is indeed sensitive to the choice of value on the equivalent sample size parameter. Due to the similarity between the MPL and the BDeu score considered in Silander et al. (2007), one would expect the MPL to display a similar behavior. In this work we primarily focus on the setting where $N = 1$, and simulations are used to demonstrate the adequacy of this choice.

An important property preferably satisfied by a scoring function is consistency. By consistency we mean that, under the assumption that the generating distribution is faithful to a Markov network structure, the score will favor the true graph when the sample size tends to infinity. The following theorem establishes that MPL is indeed a consistent scoring function for Markov networks.

Theorem 1 *Let $G^* \in \mathcal{G}$ be the true graph structure, of a Markov network over $(X_1, \ldots, X_d)$, with the corresponding Markov blankets $mb(G^*) = \{mb^*(1), \ldots, mb^*(d)\}$. Let $\theta_{G^*} \in \Theta_{G^*}$ define the corresponding joint distribution which is faithful to G^* and from which a sample $\mathbf{x}$ of size n is obtained. The local MPL estimator*

$$\widehat{mb}(j) = \underset{mb(j) \subseteq V \setminus j}{\arg\max}\ p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)})$$

is consistent in the sense that $\widehat{mb}(j) = mb^*(j)$ eventually almost surely as $n \to \infty$ for $j = 1, \ldots, d$. Consequently, the global MPL estimator

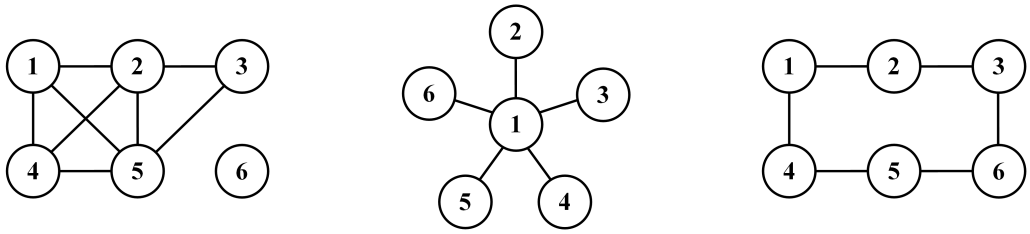
$$\hat{G} = \underset{G \in \mathcal{G}}{\arg\max}\ \hat{p}(\mathbf{x} \mid G)$$

is consistent in the sense that $\hat{G} = G^*$ eventually almost surely as $n \to \infty$.

Proof See Appendix A.

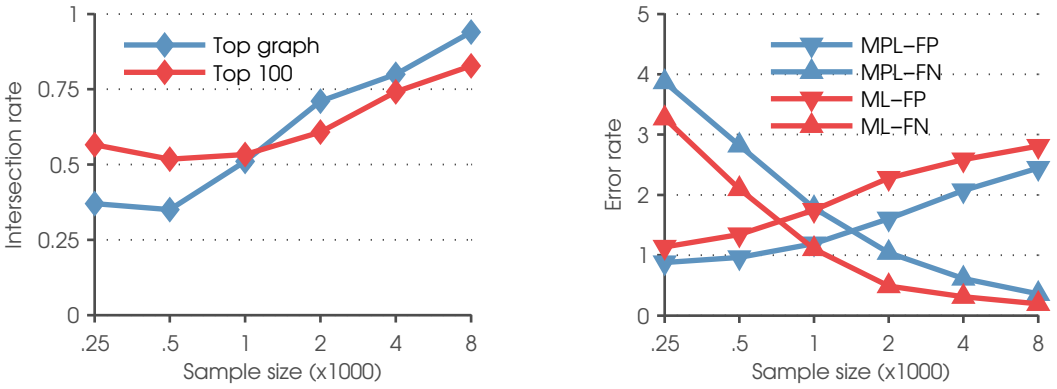
Although consistency alone is a reassuring theoretical property, it is also important to recognize its limitations. In particular, the assumptions under which the result is obtained rarely hold in practice. Therefore it is important to investigate how well the MPL performs in practice. In Section 6 we do a series of large-scale numerical simulations to investigate how well the MPL performs in combination with the search algorithms introduced in Section 5. Before that, we conduct a small-scale simulation study to gain an insight into the behavior of the MPL both when it comes to choosing the optimal graph as well as ranking the most plausible graphs.

In the first part of the experiment we restrict the model space to chordal graphs. By doing so we can calculate the ML of each considered graph and compare it to the MPL. The ML of a chordal graph is usually calculated by factorizing the likelihood according to the maximal cliques and separators of the graph (see e.g. Corander et al., 2008), however, there



(a) (b) (c)

Figure 1: Graphs used in the simulations in Section 4.2.



(a) (b)

Figure 2: Comparison of the MPL and ML graph rankings for different sample sizes. In (a) the similarity of rankings are compared for the top ranked graph and the 100 top ranked graphs. In (b) the average FP and FN rates are compared for the 100 top ranked graphs.

is an alternative approach. To any chordal graph, there exists a collection of Markov equivalent DAGs, each encoding an equivalent dependence structure as the undirected graph. We can therefore evaluate the ML of an undirected chordal graph by the BDeu metric (Buntine, 1991) of one of the equivalent DAGs. Since the BDeu metric assigns the same score to all Markov equivalent DAGs, it does not depend on which DAG being picked for evaluation. The main advantage of the DAG-based approach is that the structural form of the ML is similar to the MPL since the factorization of the likelihood and the choice of hyperparameters are done analogously. Consequently, we can apply both methods under fairly similar conditions such that the different behaviors are primarily due to different fundamental characteristics of the score functions.

First, we used the graph in Figure 1a as base for the generating model. The number of possible graphs over six nodes is $2\hat{}\binom{6}{2} = 32768$ and 18154 of these are chordal. To generate a distribution according to a graph, we assigned values to the maximal clique factors in (1) by independently sampling from a uniform distribution over $(0, 1)$. We generated ten distributions and for each distribution we generated ten samples. The final results are thus averaged over hundred samples. We performed an exhaustive evaluation of the chordal graphs and listed the hundred highest ranked graphs for the respective score.

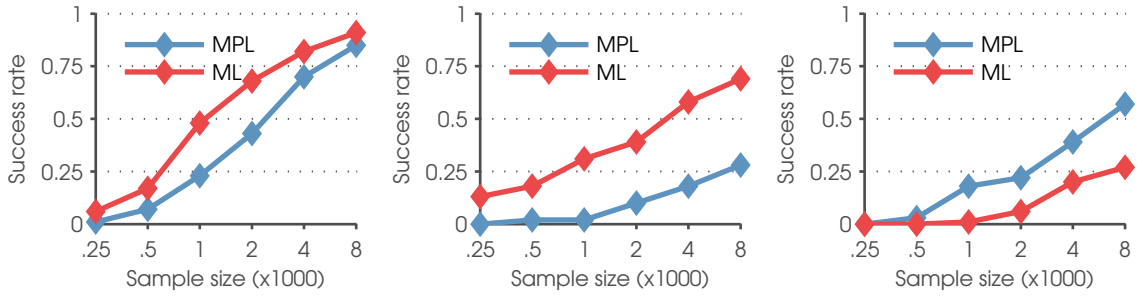


Figure 3: Comparison of the MPL and ML top ranked graph for the graphs in Figure 1. The success rate, which is plotted against the sample size, refers to the rate at which the correct graph is identified except for in (c) where the ML success rate refers to the rate at which the top ranked graph contains all the true edges.

To begin with, we consider the similarity of the rankings. Figure 2a illustrates the rate at which the ML and MPL scores agree on the top graph as well the percentage of graphs included in both of the top 100 rankings. With an increased sample size, the two scores show an increased conformity in how they rank the graphs. To further investigate the differences, Figure 2b illustrates the average rate of falsely added edges (False Positives, FPs) and falsely omitted edges (False Negatives, FNs) among the 100 top ranked graphs. Although the overall error rates are quite similar, there is a key difference between the MPL and the ML which is nicely illustrated by the figure. Since the MPL in a sense over-determines the dependence structure, it is more conservative in terms of adding edges. This phenomenon is clearly reflected by the MPL consistently having a lower false positive rate and a higher false negative rate than the ML. The difference becomes less distinct for larger sample sizes which is in concordance with Figure 2a.

One drawback of the above mentioned characteristic is that it makes the MPL less sample efficient than the ML in terms of identifying the correct graph. In Figure 3a we have plotted the rate at which the true graph was ranked as optimal by the respective score. Although the curves eventually converge for large enough sample sizes, the ML outperforms the MPL for all the considered sample sizes. This weakness is exaggerated for graphs containing large Markov blankets compared to the maximal clique sizes. As an ultimate example of this, consider the star graph in Figure 1b for which there is one hub node connected to all the other nodes. In Figure 3b we see that the ML has a clear advantage over the MPL, for this type of graph, for limited sample sizes. Still, the curves will eventually converge as confirmed by Theorem 1.

As expected, the ML is to be preferred over the MPL when it comes to picking the optimal chordal graph. However, we conclude this section by giving an example that illustrates the importance of going beyond chordal graphs, which for larger systems only make up a small fraction of the graph space. In the last experiment we also consider non-chordal graphs. In particular, we based our generating model on the non-chordal graph in Figure 1c. Since the ML can only be evaluated for chordal graphs, it cannot discover the true graph. Therefore, we change the criterion for success for the ML by looking at the true

positives. If the top ranked graph contains all of the true edges, we consider it to be correct. In this setup the MPL clearly outperforms the ML as seen in Figure 3c. This is a natural consequence considering that the ML needs to add three spurious edges in order to form a chordal graph that contains all the true edges. As in the previous case, the curves will eventually converge for large enough sample sizes. Still, a model based on a graph with spurious edges contains redundant parameters which inevitably destabilize a subsequent parameter estimation process.

4.3 Computational complexity

Whereas the computational complexity of the ML is determined by the size of the maximal cliques, the computational complexity of the MPL is determined by the size of the Markov blankets. The (log-)MPL is calculated by the sum

$$\begin{aligned}\log \hat{p}(\mathbf{x} \mid G) &= \sum_{j=1}^{d} \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}, G) \\ &= \sum_{j=1}^{d} \sum_{l=1}^{q_j} \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}^{(l)}, G) \\ &= \sum_{j=1}^{d} \sum_{l=1}^{q_j} \left[\log \Gamma(\alpha_{jl}) - \log \Gamma(n_{jl} + \alpha_{jl}) + \sum_{i=1}^{r_j} \left[\log \Gamma(n_{ijl} + \alpha_{ijl}) - \log \Gamma(\alpha_{ijl}) \right] \right],\end{aligned}$$

which consists of $\sum_{j=1}^{d} q_j(2+2r_j)$ terms. Since $r_j = |\mathcal{X}_j|$ does not depend on the graph, the number of terms, associated with a node j, is mainly determined by the number of Markov blanket configurations, q_j, which grows exponentially with the size of the Markov blanket. Thereby, the complexity of calculating the MPL of a graph is to a high extent determined by the maximal Markov blanket size. Still, it is important to note that the partial sum

$$\log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}^{(l)}, G) = \log \Gamma(\alpha_{jl}) - \log \Gamma(n_{jl} + \alpha_{jl}) + \sum_{i=1}^{r_j} \left[\log \Gamma(n_{ijl} + \alpha_{ijl}) - \log \Gamma(\alpha_{ijl}) \right],$$

does not contribute to the MPL if the corresponding Markov blanket configuration is not represented in the data. Consequently, the maximum number of terms evaluated by a non-naive implementation is $\sum_{j=1}^{d} \min(q_j, n)(2 + 2r_j)$ where n is the number of observations in the dataset. Furthermore, for a large Markov blanket of node j, the number of distinct configurations present in a dataset is, in practice, usually far less than $\min(q_j, n)$.

If we look at the MPL from an optimization perspective, it is easy to see that its variable-wise decomposition makes it a convenient candidate for search algorithms based on local changes. To compare the plausibility of two graphs, $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, we can calculate the ratio of their MPLs,

$$K(G_1, G_2) = \frac{\hat{p}(\mathbf{x} \mid G_1)}{\hat{p}(\mathbf{x} \mid G_2)},$$

or equivalently the log-ratio,

$$\log K(G_1, G_2) = \log \hat{p}(\mathbf{x} \mid G_1) - \log \hat{p}(\mathbf{x} \mid G_2),$$

which is basically the pseudo-version of log-Bayes factor or the log-Bayes pseudo-factor. Assume there is a single edge difference,

$$\{E_1 \cup E_2\} \setminus \{E_1 \cap E_2\} = \{i, j\},$$

between the graphs. This implies that $mb(i)$ and $mb(j)$ are the only Markov blankets that differ in the two graphs. Consequently, log-Bayes pseudo-factor is simply evaluated by

$$\begin{aligned}\log K(G_1, G_2) = &\log p(\mathbf{x}_i \mid \mathbf{x}_{mb(i)}, G_1) + \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}, G_1) - \\ &\log p(\mathbf{x}_i \mid \mathbf{x}_{mb(i)}, G_2) - \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}, G_2)\end{aligned}$$

since the rest of the terms cancel each other out.

4.4 Related work

In addition to the asymptotically equivalent PIC (see proof of Theorem 1) by Csiszár and Talata (2006), the MPL is very closely related to a class of models known as dependency networks (Heckerman et al., 2001). In fact, the general concept of using pseudo-likelihood for Markov networks has an alternative interpretation in terms of this class of models.

The distribution of a dependency network is, like a Bayesian network, represented by variable-wise conditional distributions in a pseudo-likelihood type manner. The directed graph of a dependency network may thus, in contrast to a Bayesian network, contain cycles. A dependency network does not in general represent a consistent distribution in the sense that the local distribution cannot be inferred from a joint distribution over all the variables. Consequently, one must rely on Gibbs sampling to perform inference in such models. In contrast to dependency networks, Markov networks always represent a consistent distribution. Still, any Markov network with the undirected graph G can be represented by a consistent dependency network with a symmetric directed graph containing the same structural adjacencies as G (Theorem 1 & 4 Heckerman et al., 2001).

The obvious advantage of dependency networks in terms of structure learning is that the local structure of each node can be learned independently of the rest of the network. The local structures can be inferred using a variety of regression-based techniques. In particular, Heckerman et al. (2001) model the local structures using probabilistic decision trees in conjunction with a Bayesian score originally derived by Friedman and Goldszmidt (1996) for the purpose of including context-specific independence in the learning process of Bayesian networks. Lowd and Davis (2014) converted this approach into a feature selection method by transforming the trees into features of a Markov network. The authors mention the risk of overfitting by generating long specialized features. Due to the feature-edge relation described in Section 2, the implication of such overfitting would be emphasized in terms of graph structure discovery.

The logistic regression approach of Ravikumar et al. (2010) is another method that has a natural interpretation in terms of the dependency network framework. The solutions of the separate regression problems represent a structure of a general dependency network and must be made symmetric in order to be consistent with a structure of a Markov network. In contrast, the problem formulation in the closely related approach by Höfling and Tibshirani (2009) ensures that the network is kept symmetric and even consistent during the optimization process.

Our MPL can be interpreted as the ML of a symmetric dependency network under the standard conditional parameterization defined in (8). Without the parameter independence assumption, the MPL would correspond to the ML of a consistent dependency network under our parameterization. This would make up a very natural option for objective function if it could be evaluated efficiently. Still, the goal of the MPL is to learn a graph structure rather than a specific model. If merely considering the dependence structure, each graph of a Markov network has an equivalent counterpart in terms of a symmetric dependency network structure. Therefore it makes sense to enforce consistency among the Markov blankets, that is, to only consider symmetric dependency networks. In contrast to a general dependency network, the MPL evaluates the inclusion of an edge $\{i, j\}$ by comparing the potential benefit from adding j to $mb(i)$ against the potential loss of adding i to $mb(j)$.

5. MPL optimization

The straightforward global MPL-based optimization problem is formulated by

$$\arg\max_{G \in \mathcal{G}} \ \log \hat{p}(\mathbf{x} \mid G) + \log p(G) \tag{10}$$

where $p(G)$ is the graph prior distribution which can account for any prior belief regarding for example the degree of sparsity. To maintain the useful structure of the MPL, the prior must follow a similar decomposition. This is achieved by defining the prior in terms of mutually independent prior beliefs on the individual Markov blankets. In Section 6.3 we give an example of such a prior, however, in the remainder of the section we assume a uniform prior and the term $p(G)$ is therefore omitted. Still, the methods presented in this section are also directly applicable under any prior that follows the same decomposition as the MPL.

Due to rapidly growing size of the discrete optimization space, the global optimization problem (10) is clearly intractable already for moderate-sized systems. Hence, we need to construct an algorithm that finds approximate solutions of satisfactory quality in a reasonable time. To ensure applicability in a genuinely high-dimensional setting, the algorithm is designed to exploit the structural decomposition of the MPL by breaking down the problem into two steps instead of directly approaching the global problem (10).

Since each graph G is uniquely specified by its collection of Markov blankets $mb(G) = \{mb(j)\}_{j=1}^{d}$, we can reformulate (10) as

$$\begin{gathered} \arg\max_{mb(G) \in \times_{j \in V} \mathcal{P}(V \setminus j)} \ \sum_{j=1}^{d} \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}) \\ \text{subject to} \quad i \in mb(j) \Rightarrow j \in mb(i) \text{ for all } i, j \in V \end{gathered} \tag{11}$$

where $\mathcal{P}(V \setminus j)$ is the power set of $V \setminus j$ representing all possible Markov blankets of node j. From (11) it is easy to see that our problem is basically made up of d dependent subproblems that are connected through the consistency constraint. By omitting the constraint we remove the dependence among the subproblems and obtain the relaxed problem

$$\arg\max_{mb(G) \in \times_{j \in V} \mathcal{P}(V \setminus j)} \ \sum_{j=1}^{d} \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}). \tag{12}$$

Since the d subproblems now are independent of each other, we can finally reformulate (12) by breaking it down into a collection of stand-alone Markov blanket discovery problems,

$$\underset{mb(j)\subseteq V\setminus j}{\arg\max} \ \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}) \quad \text{for } j = 1, \ldots, d, \tag{13}$$

which can be solved completely in parallel considerably improving real time efficiency. Since each individual subproblem in itself is still intractable, in Section 5.1 we introduce an efficient deterministic search algorithm that gives an approximate solution.

The relaxation step shifts the focus from the strictly score-based view in (10) towards a constraint- or regression-based view, or in terms of dependency networks, from symmetric to general. It is worth noticing that the consistency result established in Theorem 1 still holds under the relaxed problem formulation.

By solving the relaxed problem we usually obtain a solution inconsistent with a Markov network structure. We could simply post-process the solution using either a $\wedge$ (and) criterion,

$$E_{\wedge} = \{\{i, j\} \in \{V \times V\} : i \in mb(j) \wedge j \in mb(i)\}$$

or a $\vee$ (or) criterion,

$$E_{\vee} = \{\{i, j\} \in \{V \times V\} : i \in mb(j) \vee j \in mb(i)\}.$$

These criteria are quite standard among constraint- or regression-based methods, however, neither of them is quite satisfactory from an MPL optimization perspective. Therefore we propose a second optimization phase whose goal is to combine the inconsistent Markov blankets from the first phase into a coherent structure which is MPL-optimal on a reduced model space determined by the relaxed solution.

More specifically, the edge set in $E_{\vee}$ is considered to be the result of a prescan that identifies eligible edges. The original problem (10) is then solved with respect to the reduced model space $\mathcal{G}_{\vee} = \{G \in \mathcal{G} : E \subseteq E_{\vee}\}$, that is

$$\underset{G\in\mathcal{G}_{\vee}}{\arg\max} \ \log \hat{p}(\mathbf{x} \mid G).$$

The reduced model space $\mathcal{G}_{\vee}$ is in general considerably smaller than $\mathcal{G}$. In Section 5.2 we discuss a method that under certain circumstances can solve the above problem exactly. In Section 5.3 we describe a fast deterministic approximate algorithm that can be applied also in situations when the exact method is infeasible.

5.1 Local Markov blanket discovery using greedy hill climbing

To solve the relaxed problem, we basically need a Markov blanket discovery algorithm whose goal is to optimize the local MPL for each node independently of the solutions of the other nodes. For this we use an approximate deterministic hill climbing procedure similar to the interIAMB algorithm by Tsamardinos et al. (2003).

An outline of the algorithm is presented in Algorithm 1 and the general idea is as follows. The algorithm is based on the two basic operations by which members are added to or deleted from the Markov blanket. The method is initiated with the empty Markov blanket and all other nodes are considered potential Markov blanket members. At each

Algorithm 1 Procedure for optimizing the local MPL of a node using greedy hill climbing.

Procedure Markov-Blanket-Hill-Climb(
j, //*Current node*
$\mathbf{x}$, //*Complete dataset*
)

$mb(j), \widehat{mb}(j) \leftarrow \varnothing$
while $\widehat{mb}(j)$ has changed
  $C \leftarrow V \setminus \{mb(j) \cup j\}$
  $mb(j) \leftarrow \widehat{mb}(j)$
  for each $i \in C$
    if $\log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)\cup i}) > \log p(\mathbf{x}_j \mid \mathbf{x}_{\widehat{mb}(j)})$
      $\widehat{mb}(j) \leftarrow mb(j) \cup i$
    end
  end
  while $\widehat{mb}(j)$ has changed & $|\widehat{mb}(j)| > 2$
    $mb(j) \leftarrow \widehat{mb}(j)$
    for each $i \in mb(j)$
      if $\log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)\setminus i}) > \log p(\mathbf{x}_j \mid \mathbf{x}_{\widehat{mb}(j)})$
        $\widehat{mb}(j) \leftarrow mb(j) \setminus i$
      end
    end
  end
end
return $\widehat{mb}(j)$

iteration it adds to the Markov blanket the node that induces the greatest improvement to the local MPL and updates the set of potential members accordingly. When the size of the Markov blanket grows larger than two, the algorithm interleaves each successful addition-step with a deletion phase. In the deletion phase, the algorithm removes the node that induces the largest improvement to score. The deletion-step is repeated until removal of a node no longer increases the score or the size of the Markov blanket is smaller than three. When the addition-phase is iterated through without a successful addition, a local maximum has been reached, the algorithm terminates and returns the identified Markov blanket.

To examine the computational complexity of the algorithm, we consider the cost of performing an complete iteration where mb denotes the current Markov blanket. In the addition phase, each of the $d - 1 - |mb|$ candidate members needs to be evaluated by calculating the local MPL for Markov blankets of size $|mb| + 1$. Say that a node is added to the Markov blanket which is now of size $|mb| + 1$. In the first iteration of a potential deletion phase, the removal of each of the $|mb| + 1$ Markov blanket members is evaluated by calculating the local MPL for Markov blankets of size $|mb|$. In practice, the most lately added node can be skipped in the first iteration. In a potential successive deletion step, $|mb|$ Markov blankets of size $|mb| - 1$ must be evaluated and so on.

The recurring deletion-phase of the algorithm attempts to keep the size of the Markov blanket as small as possible during the search in order to improve both sample and time efficiency. Still, the computational cost of the method is strongly dependent on the size of identified Markov blanket since a large Markov blanket naturally requires many iterations. Furthermore, an iteration becomes more expensive as the current Markov blanket grows larger since the cost of evaluating the local MPL is highly dependent on the size of the Markov blanket (see Section 4.3).

5.2 Global graph discovery using pseudo-boolean optimization

There has recently been a considerable interest in use of computational logic algorithms for structure learning of both Bayesian and Markov networks (Cussens, 2008; Bartlett and Cussens, 2013; Corander et al., 2013; Berg et al., 2014; Parviainen et al., 2014). In this section we describe how

$$\arg\max_{G \in \mathcal{G}_\vee} \log \hat{p}(\mathbf{x} \mid G) \tag{14}$$

can be cast as a pseudo-boolean optimization (PBO) problem (Boros and Hammer, 2002) which can be solved by existing mixed integer programming solvers such as the SCIP solver (Berthold et al., 2009; Achterberg, 2009).

A PBO problem consists of an objective function and a set of (in)equality constraints over boolean variables. To formulate our optimization problem as a PBO problem, we need to introduce two types of propositional variables:

1. **Edge variables**: For each edge $\{i, j\} \in E_\vee$, a variable $x_{\{i,j\}}$ is introduced. If the value of $x_{\{i,j\}}$ is 1 (true) in a solution, the associated edge is included in the graph. If the value of $x_{\{i,j\}}$ is 0 (false), the associated edge is not included in the graph.

2. **Markov blanket variables**: Each node j is associated with a set of candidate Markov blankets defined as all subsets of $mb_\vee(j)$ which is the Markov blanket of node j in $G_\vee$. Let d_j be the number of nodes in $mb_\vee(j)$. The Markov blanket candidates are denoted by $mb_k(j)$ for $k = 1, \ldots, m_j$ where $m_j = 2^{d_j}$. For each candidate Markov blanket, a variable $x_{mb_k(j)}$ is introduced. If the value of $x_{mb_k(j)}$ is 1 (true) in a solution, the Markov blanket of node j is equal to $mb_k(j)$ in the graph. If the value of $x_{mb_k(j)}$ is 0 (false), the Markov blanket of node j is not equal to the k:th candidate.

Each complete instantiation of the edge variables will correspond to a distinct graph in the considered graph space. The purpose of the edge variables is to ensure that the combined Markov blankets correspond to a coherent graph structure. Consequently, we need to connect the edge variables to the blanket variables in such a way that the value of blanket variable is true if and only if all edge variables associated with edges induced by the Markov blanket are true and the remaining edge variables are false. More formally, we need to introduce a constraint corresponding to the propositional formula

$$x_{mb_k(j)} \leftrightarrow (x_{\{v_1,j\}} \wedge x_{\{v_2,j\}} \wedge \cdots \wedge x_{\{v_l,j\}} \wedge \neg x_{\{v_{l+1},j\}} \wedge \neg x_{\{v_{l+2},j\}} \wedge \cdots \wedge \neg x_{\{v_{d_j},j\}}) \tag{15}$$

where

$$\{v_1, \ldots, v_l\} = mb_k(j) \text{ and } \{v_{l+1}, \ldots, v_{d_j}\} = mb_\vee(j) \setminus mb_k(j).$$

If we now consider the variables taking on values 0 and 1 rather than false and true, the above formula can be expressed as the pseudo-boolean equality constraint

$$x_{mb_k(j)} - \left(\prod_{i=1}^{l} x_{\{v_i,j\}}\right)\left(\prod_{i=l+1}^{d_j} \bar{x}_{\{v_i,j\}}\right) = 0, \tag{16}$$

where $\bar{x}_{\{v_i,j\}} = 1 - x_{\{v_i,j\}}$. For any assignment of the variables, it is clear that the value of formula (15) is true if and only if constraint (16) is satisfied. For each node and Markov blanket candidate, we add constraint (16) to the PBO problem. This will ensure that any feasible instantiation of the introduced variables must coincide with a graph structure of a Markov network.

The constraints expressed by equation (16) are sufficient on their own, however, to facilitate the optimization process we also introduce the following constraint for each node:

$$\sum_{k=1}^{m_j} x_{mb_k(j)} = 1 \tag{17}$$

By including constraint (17), we explicitly require that exactly one candidate is selected for each node. Even though this is already implied by constraints (16), the implication is not straightforward since the candidate variables are related to each other via the edge variables. Consequently, to realize that any two given candidate variables of the same node can not be true simultaneously, some of the edge variables need to be assigned. Therefore, including constraint (17) helps the solver to tighten the bounds of the feasible region (and objective function) early on.

Finally, we need to express our objective function. For this we introduce the Markov blanket candidate weights

$$w(j,k) = -\lfloor K \cdot \log p(\mathbf{x}_j | \mathbf{x}_{mb_k(j)}) \rfloor$$

where K is a large positive integer. As required in a PBO objective function, the floor function transforms the weights into integers. The objective function to be minimized can now be expressed by

$$\sum_{j=1}^{d}\sum_{k=1}^{m_j} w(j,k) \cdot x_{mb_k(j)}. \tag{18}$$

With a large enough K, the solution to the PBO problem

$$\underset{\substack{mb(j)\subseteq mb_\vee(j)\\ j=1,\ldots,d}}{\arg\min} \sum_{j=1}^{d}\sum_{k=1}^{m_j} w(j,k) \cdot x_{mb_k(j)}, \tag{19}$$

subject to constraint (16) (and (17)), is equivalent to the solution to the optimization problem

$$\underset{\substack{mb(j)\subseteq mb_\vee(j)\\ j=1,\ldots,d}}{\arg\min} \; -K \sum_{j=1}^{d} \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}), \tag{20}$$

Algorithm 2 Procedure for optimizing the MPL using greedy hill climbing.

Procedure Graph-Hill-Climb(
$\mathcal{G}_\vee$ //*The considered graph space*
$\mathbf{x}$, //*Complete dataset*
)

$G, \widehat{G} \leftarrow \varnothing$
while $\widehat{G}$ has changed
  $G \leftarrow \widehat{G}$
  for each $G' \in \mathcal{N}_{\mathcal{G}_\vee}(G)$
    if $\hat{p}(\mathbf{x} \mid G') > \hat{p}(\mathbf{x} \mid \widehat{G})$
      $\widehat{G} \leftarrow G'$
    end
  end
end
return $\widehat{G}$

subject to the constraint in (11), which in turn is equivalent to the original optimization problem (14).

The obvious advantage of this approach is that we are guaranteed to obtain the exact (or global) solution to the problem. However, the method can only be applied in certain situations since the number of variables and constraints for each node grows exponentially with the number of the potential Markov blanket members. More specifically, the total number of introduced boolean variables is $|E_\vee| + \sum_{j=1}^{d} 2^{d_j}$ and the total number of introduced equality constraints is $d + \sum_{j=1}^{d} 2^{d_j}$. In addition, the weight of each candidate must be calculated and stored prior to the actual optimization. Consequently, the feasibility of this approach depends strongly on the sizes of the Markov blankets in $G_\vee$. Hence, in the next section we also introduce an alternative approximate algorithm that can be applied also in intractable situations.

5.3 Global graph discovery using greedy hill climbing

The variable-wise factorization of the MPL makes it particularly well-suited for global search algorithms based on local changes. As a stochastic option, the non-reversible MCMC-based approach by Corander et al. (2008) is directly applicable for MPL optimization. Here we propose a simple deterministic approach in form of a greedy hill climbing (HC) algorithm which has also been used for learning Bayesian networks (see e.g. Heckerman et al., 1995). Local edge change algorithms move between neighboring graph structures during the optimization procedure. The set of neighbors of a graph G in a graph space $\mathcal{G}$ is denoted by $\mathcal{N}_{\mathcal{G}}(G)$ and defined as all graphs in $\mathcal{G}$ that can be reached from G by a adding or removing a single edge.

An outline of the algorithm is presented in Algorithm 2 and the general idea is as follows. The empty graph is set as the initial graph and the considered optimization space is $\mathcal{G}_\vee$. At each iteration, all neighbors of the current graph are evaluated. At the end of the iteration, we choose the highest scoring graph from the neighbors, assuming that it has a higher score

than the current graph, and repeat the procedure. If no candidate among the neighbors has a higher score than the current graph, a local maximum has been reached, the algorithm terminates and returns the identified graph.

We examine the computational complexity of the proposed algorithm by considering the calculations required at each iteration. The cost of evaluating the specified expressions was discussed in Section 4.3. We show that, by implementing smart caching, the efficiency of the algorithm can be improved considerably. Let G_t be the current graph at iteration t and let $G'_t \in \mathcal{N}_{\mathcal{G}_\vee}(G_t)$ differ with respect to the edge $\{i, j\}$. To compare G'_t to G_t we calculate the log-Bayes pseudo-factor

$$\begin{aligned}\log K(G^*_t, G_t) = &\log p(\mathbf{x}_i \mid \mathbf{x}_{mb(i)}, G'_t) + \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}, G'_t) - \\ &\log p(\mathbf{x}_i \mid \mathbf{x}_{mb(i)}, G_t) - \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}, G_t).\end{aligned}$$

The above expression must be evaluated for each candidate in the neighbor set $\mathcal{N}_{\mathcal{G}_\vee}(G_t)$ which has a maximum cardinality of $\binom{d}{2}$ if all edges are included. However, since

$$\log p(\mathbf{x}_i \mid \mathbf{x}_{mb(i)}, G_t) \text{ and } \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}, G_t)$$

are determined by the current graph, they can be stored and re-used. Consequently,

$$\log p(\mathbf{x}_i \mid \mathbf{x}_{mb(i)}, G'_t) \text{ and } \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}, G'_t)$$

are the only terms that specifically need to be calculated to evaluate the neighbor associated with the edge change $\{i, j\}$.

In the first iteration, d local MPLs with empty Markov blankets must be evaluated. Additionally, each possible neighbor must be evaluated by calculating two local MPLs with Markov blanket size one. In subsequent iterations, we can further exploit the decomposition of the MPL by noting that most of the log-factors from the previous iteration remain unchanged under the new current graph. In fact, the only edge changes that need to be re-evaluated are those that overlap with the previous change. Say that the current graph G_t was attained by adding (deleting) the edge between $\{k, l\}$ to (from) G_{t-1}. In line with earlier notation, let G'_{t-1} be the neighbor of G_{t-1} that differs with respect to the edge $\{i, j\}$. Given the context, we now have that

$$\log K(G'_t, G_t) = \log K(G'_{t-1}, G_{t-1}) \text{ if } \{i, j\} \cap \{k, l\} = \varnothing.$$

Consequently, after the initial iteration it suffices to re-evaluate only a small fraction of the updated neighbor set. In the worst case, $2(d-1)$ neighbors need to be evaluated even though the maximum cardinality of the neighbor set of interest is $\binom{d}{2} - 1$ when excluding the graph from the previous iteration.

Under this optimization strategy, the MPL method is similar in spirit to the max-min hill climbing algorithm for learning Bayesian networks by Tsamardinos et al. (2006). The main difference is that both phases of our algorithm are derived from the notion of maximizing a single underlying score.

Network	Grid	Hub	Loop	Clique
Number of nodes	16	16	16	16
Number of edges	24	15	19	20
Average Markov blanket size	3.25	1.88	2.38	2.5
Maximum Markov blanket size	4	8	4	4
Chordal	No	Yes	No	Yes

Table 1: Properties of the graph components in Figure 4.

6. Experimental results

The main focus of this section is to empirically investigate the performance of the MPL using the optimization algorithms from the previous section. To evaluate our approach in a controlled setting, we compare it to other competitive methods on synthetic models as well as real-world networks. Since the graph structures of the generating models are known, it allows for a straightforward and fair assessment of the algorithms. To finally illustrate the potential of our method, we also present a real high-dimensional knowledge discovery problem on which our method is applied.

When the true structure of the generating network is known, the quality of an output network is readily assessed by the number of errors in terms of FPs and FNs. As our main measure of quality we consider the sum of FPs and FNs which is the Hamming distance between the output and true network. Consequently, a low value on the Hamming distance corresponds to structural resemblance to the true network and the minimum value of zero is obtained for the correct graph. In addition to structural resemblance, we monitor the execution times for the different methods.[1] The total runtimes of all algorithms are reported along with the maximum discovery time for a single Markov blanket. The maximum Markov blanket discovery time would be the total real time required if the local problems were solved in parallel rather than in serial fashion.

If not otherwise mentioned, we set the equivalent sample size parameter $N = 1$ which results in a weak parameter prior. For the main part of the experiments, we set $p(G)$ to be uniform since we want to investigate how well the MPL alone performs as a metric for graph structures.

6.1 Synthetic Markov networks

In this section we use synthetic models to generate datasets of different sizes to systematically compare the performance of the MPL combined with our optimization algorithms. Moreover, we compare the MPL against different competing methods for structure learning of Markov networks. For simplicity, we restrict the synthetic networks to be made up of binary variables.

The synthetic graphs were formed by combining disconnected components in form of the four 16-node graphs illustrated in Figure 4. These graphs represent different structural characteristics present in realistic models and some of their properties are listed in Table 1. In particular, as already shown in Section 4.2, the hub network in Figure 4b represents

1. All experiments were carried out in Matlab except for the PBO, which was solved using the SCIP solver (web site: `http://scip.zib.de/`). The experiments were performed on a standard PC architecture with 2.66GHz dual-core processors.

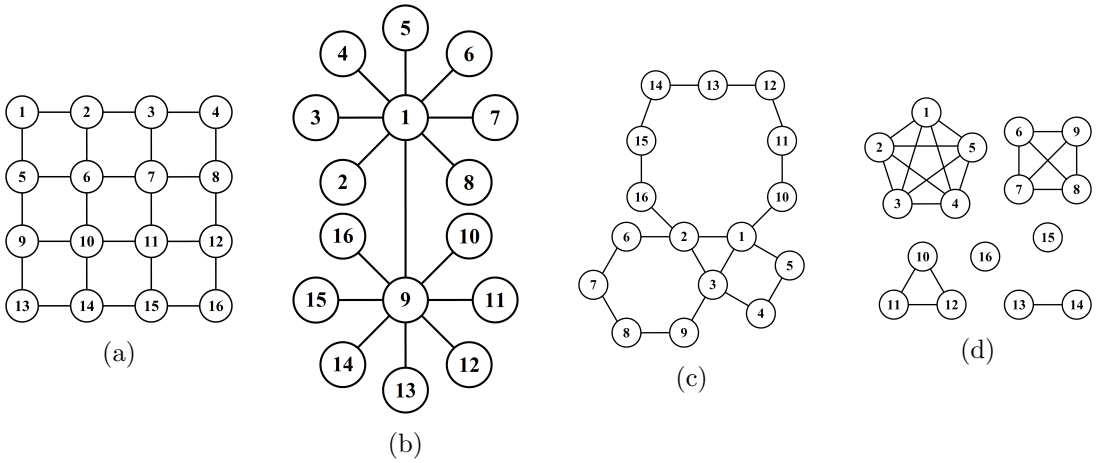
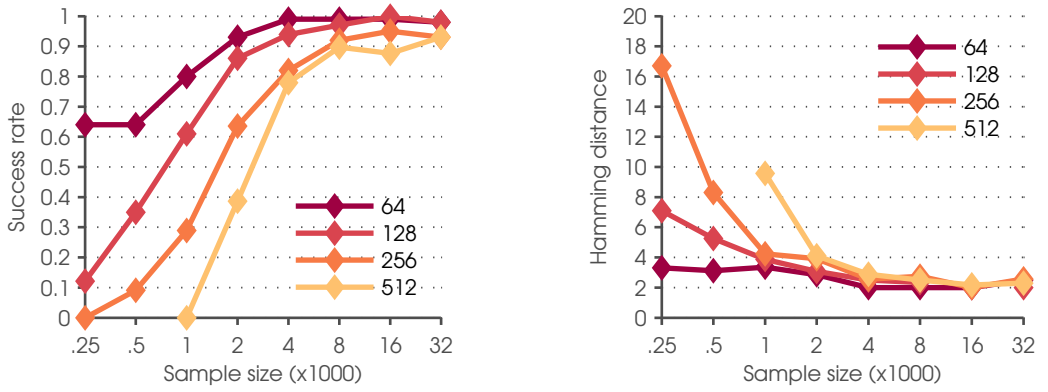


Figure 4: Synthetic graphs used in Section 6.1: (a) Grid, (b) Hub, (c) Loop, and (d) Clique.

a structural characteristic that is especially hard to capture for the MPL even though it is a rather simple tree structure. Initially, one replica of each subgraph was combined to form a structure over 64 variables. This procedure was then repeated with 2, 4, and 8 replicas to form network structures over 128, 256, and 512 variables, respectively. Each final network structure thus contained all the structural characteristics present in the graph components. The advantage of this approach is that the disconnected nature of the generating networks facilitates the sampling procedure substantially since each distinct subnetwork can be sampled directly from its corresponding joint distribution independently of the rest of the network. In practice, a distribution was generated by randomly sampling the maximal clique factors in (1). Each factor value $\phi(x_C)$ was drawn, independently of the other values, from a uniform distribution over $(0, 1)$. Consequently, the strength of the dependencies entailed by the edges may have varied considerably. To increase the stability of our results, for each sample size and graph structure, we generated 10 distributions from each of which 10 datasets were sampled. In total, under each setup, we learned 100 model structures over which the final results were averaged. The experiments were performed for sample sizes ranging from 250 to 32000.

First we examine how our approximate algorithm, HC, performs in comparison with our exact algorithm, PBO, in the second phase of the optimization process. Since the exact method finds the globally MPL-optimal graph on the reduced graph space $\mathcal{G}_\vee$, we can use it as a gold standard to which we compare our approximate method. Since the feasibility of the exact method is restricted by the output of the first phase, we need to filter out instances that are not solvable in a reasonable time. We restricted the comparison to instances where the total number of Markov blanket candidates is less than 15000. We also set a time limit on the solver to 3600 seconds per instance. The following results are thus based on the remaining solved instances (see Table 3 for more details).

In Figure 5a we have plotted the rate at which the two methods discover identical solutions. In terms of the HC method, this corresponds to the rate at which the algorithm succeeds in reaching the global optimum. As expected, the success rate grows with an increased sample size. When given more observations the algorithm is more firmly guided



(a) (b)

Figure 5: Comparison of HC and PBO for different model sizes. In (a) the rate at which the two methods reach identical solutions is plotted against the sample size. In (b) the average Hamming distance between non-identical solutions is plotted against the sample size.

towards the optimal solution. In addition, the size of the optimization space is in general smaller for larger sample sizes due to an increased conformity among the Markov blankets from the initial phase. Overall, the HC method performs very well in identifying the optimal solution for reasonable sample sizes. We now consider the instances where the optimal solution was not reached by the HC method. In Figure 5b we have plotted the average Hamming distance between the HC solution and the optimal solution for instances when the two graphs were different. For all of the considered model sizes, the curves quickly converge towards a Hamming distance of two which is the closest a local maximum can be to the global maximum under our definition of neighboring graphs. As expected, the approximate and exact solutions tend to resemble each other somewhat less for the more extreme "large d, small n"-setups.

In the second part of the experiment, we compare the MPL against other structure learning methods that are also applicable in high dimensions. We limit the MPL approach to the less computationally expensive HC algorithm, which we from now on simply refer to as the MPL method. The other methods used in the comparison are the following:

- **PIC**: The PIC criterion by Csiszár and Talata (2006) is applied using the exact same search technique as for the MPL method. From the proof of Theorem (1), we know that MPL and PIC are asymptotically equivalent estimators, but here we examine how they perform in practice for limited sample sizes.

- **CMI**: We apply the Markov blanket discovery approach of Tsamardinos et al. (2003) who use conditional mutual information[2] (CMI) to assess if two variables are conditionally independent given some set of variables. For a fair comparison against the MPL method, we use the CMI measure combined with Algorithm 1. To form the final

2. To calculate the conditional mutual information, the Matlab package of Peng (2007, Accessed 2013-10-14) was used.

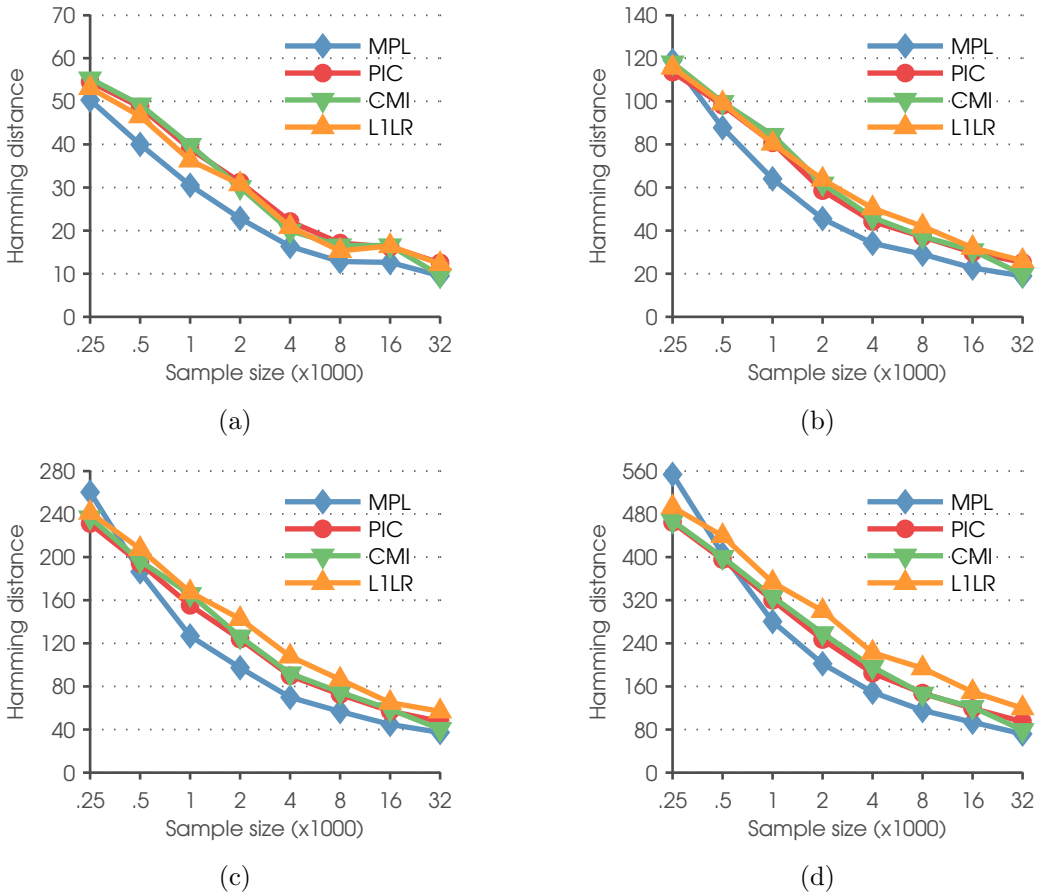


Figure 6: Comparison of the MPL method with the other structure learning methods. The average Hamming distance between the induced and true graph is plotted against the sample size for synthetic models of size (a) $d = 64$, (b) $d = 128$, (c) $d = 256$, and (d) $d = 512$.

graph, we apply either the AND or the OR criterion depending on which one results in a graph closer to the true graph in terms of Hamming distance.

- **L1LR**: We apply the L_1-regularized logistic regression[3] (L1LR) approach of Ravikumar et al. (2010) which is directly applicable on our models since we have restricted the experiments to binary variables. To form the final graph, we apply either the AND or the OR criterion depending on which one results in a graph closer to the true graph in terms of Hamming distance.

An issue with both the CMI and L1LR method is that they require the user to specify a crucial tuning parameter in form of a threshold value and a regularization weight, respectively. Consequently, to circumvent this problem, the methods were executed for the

3. The L_1-regularized logistic regression was performed using the Matlab package of Schmidt (2013, Accessed 2013-10-14).

Network	Alarm	Insurance	Hailfinder	Barley
Number of nodes	37	27	56	48
Number of edges (DAG)	46	52	66	84
Number of edges (moral graph)	65	70	99	126
Number of parameters	509	984	2656	114005
Average Markov blanket size	3.51	5.19	3.54	5.25
Maximum Markov blanket size	8	10	17	13
Average variable cardinality	2.84	3.30	3.98	8.77
Maximum variable cardinality	4	5	11	67

Table 2: Properties of the real-world Bayesian networks used in Section 6.2.

following ranges of values:

$$\lambda_{\text{CMI}} \in \{0.25, 0.1, 0.075, 0.05, 0.025, 0.01, 0.0075, 0.005, 0.0025, 0.001, 0.00075, 0.0005\},$$

$$\lambda_{\text{L1LR}} \in \{4, 6, 8, 12, 16, 24, 32, 48, 64, 96, 128, 192, 256, 384, 512, 768, 1024\}.$$

This resulted in a range of overly sparse to overly dense graphs from which we picked the graph (and parameter value), that minimized the Hamming distance, as the final solution. Since the Hamming distance tend to steadily increase when moving away from the optimal parameter value, the above range of values were chosen such that the picked parameter values (see Table 8) would lie strictly between the smallest and largest value. In a sense, the MPL also requires us to choose a value on the equivalent sample size parameter N. However, in these experiments we have fixed $N = 1$ whereas the other methods are tuned with respect to the true graph in order to perform optimally given the range of parameter values.

The results of the simulations are summarized in Table 4 and 5. In Figure 6 the average Hamming distance is illustrated for the different methods and model sizes. Overall, the MPL method performed highly satisfactorily and was marginally inferior to the other methods only for some of the most extreme "large d, small n"-settings. It is difficult to say whether this is due to the MPL itself or the approximation made by the HC method. In terms of speed (see Table 5), the MPL method was at a comparable level for all of the considered models. Furthermore, the task of determining the tuning parameter experimentally would significantly increase the total runtimes of the CMI and L1LR method.

6.2 Real-world Bayesian networks

In this section we proceed to a more realistic setting by conducting experiments on well-known real-world models, from the related class of Bayesian networks, in a similar fashion as Bromberg et al. (2009). The considered models are commonly used as benchmarks in research and are available from a number of sources[4]. To transform the directed acyclic graph of a Bayesian network into a corresponding undirected graph of a Markov network, a two-step procedure known as moralization is used (see Lauritzen, 1996; Koller and Friedman,

4. The networks used in this work were obtained from the Bayesian network repository at `http://www.bnlearn.com/bnrepository/` (Accessed 2014-08-07) and sampled using the R package of Scutari (2010).

2009). In the first step all parents of a common child are connected by an undirected edge if not already connected. In the second step the graph is made undirected by removing the direction of all directed edges. Although the local Markov property remains valid in the transformed network, some conditional independencies are lost in the moralization process due to the added edges. Consequently, the associated distribution is no longer faithful to the undirected graph making the graph identification more challenging.

We selected the four medium-sized networks which are listed along with some of their properties in Table 2. Compared to the relatively simple and balanced synthetic networks in Section 6.1, these models are more challenging due to their higher edge density and larger Markov blankets. In addition, large variable cardinalities also tend to have a negative effect on the learning time for methods such as the MPL. As before, we sampled each network for sample sizes ranging from 250 to 32000. For each network and sample size, we generated 100 samples over which the final results were averaged. We applied the same methods as in the previous section except for L1LR which without modifications is restricted to binary variables. In order to have a sufficient range of threshold values for the CMI method we added

$$\{0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5\}$$

to the range of values from the previous section.

The results of the simulations are summarized in Table 6 and 7. In Figure 7 the average Hamming distance from the moralized graph is illustrated for the different methods and networks. Again, the MPL method displayed an overall stable and good performance when compared to the other methods. However, the Hailfinder network exposes the main weakness of the MPL (and the related PIC when consistency is enforced). Although the majority of the nodes in the moral graph have relatively small Markov blankets, there is one node with a Markov blanket of size 17. As shown earlier, the MPL struggles with large Markov blankets or so-called hub nodes. As seen in Table 6, the CMI-AND method naturally suffers from the same problem, however, the CMI-OR method can to some extent circumvent it. In terms of speed (see Table 7), the MPL is the slowest of the considered methods, however, its runtimes are still at a reasonable level considering the performance. A reason for the slower runtimes is that the MPL method tends to produce denser graphs than the PIC as well as the CMI method when tuned with respect to the Hamming distance. Consequently, the MPL method requires more iterations which will affect the runtimes negatively, especially when the variable cardinalities are increased. However, it should be noted that the choice of threshold value by cross-validation would again significantly increase the computation time for CMI method.

All MPL simulations this far have been performed under the fixed equivalent sample size $N = 1$. Whereas λ_{CMI} and λ_{L1LR} have a rather clear interpretation in terms of their effect on the graph, the effect of N is not as easy to interpret. Silander et al. (2007) showed experimentally that the maximum a posteriori (MAP) structure of the BDeu score for Bayesian networks is sensitive to the choice of N. Furthermore, they noted that larger values of N tend to produce denser MAP graphs. Since the MPL and BDeu share the same basic structure, one would expect to see a similar behavior between the two scores. To investigate this we conclude this section by performing an additional simulation study for the Alarm network for

$$N \in \{1, 4, 8, 16, 32, 64, 128, 256\}.$$

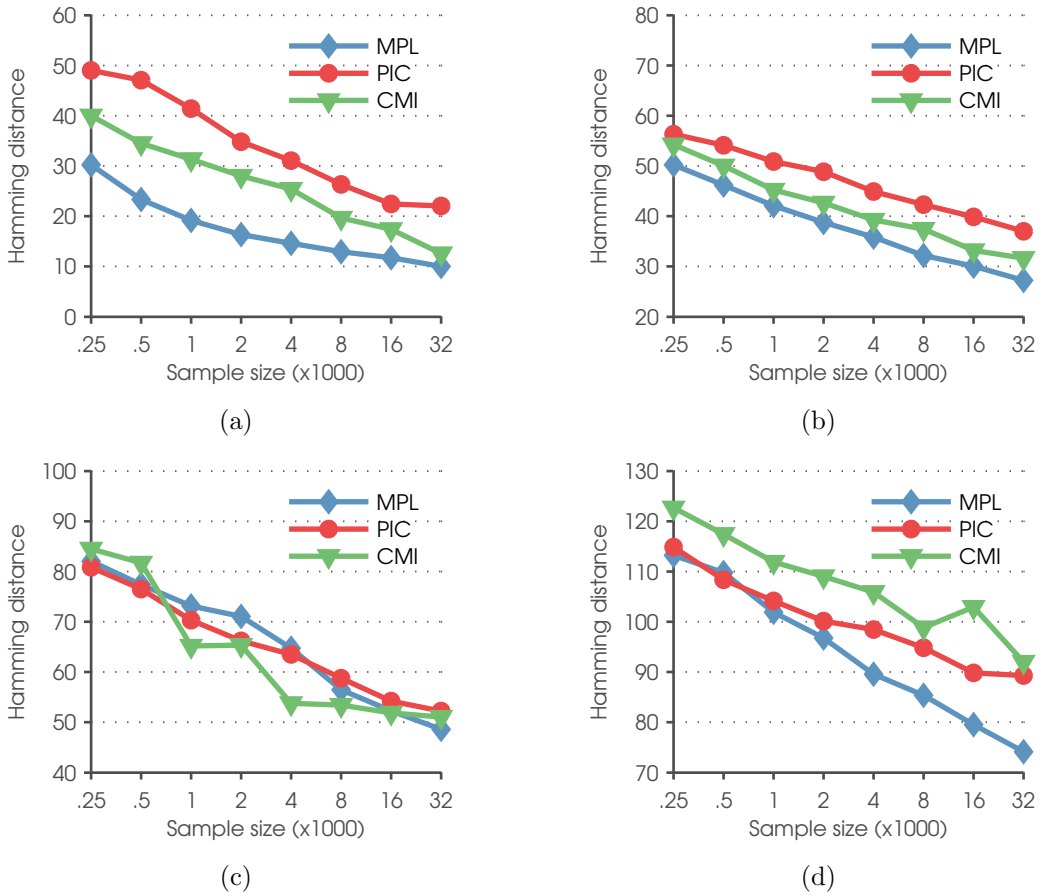


Figure 7: Comparison of the MPL method with the other structure learning methods. The average Hamming distance between the identified and true moral graph is plotted against the sample size for the (a) Alarm, (b) Insurance, (c) Hailfinder, and (d) Barley network.

The results of the simulation are summarized in Table 6b and the Hamming distances are illustrated in Figure 8. As expected, the results indicate a similar behavior as the BDeu metric in the sense that the MPL method produced denser graphs for larger values of N. Furthermore, the results in Table 6b also indicate that larger values of N can be beneficial for larger samples, however, $N = 1$ appears to be a reasonable choice when considering the complete range of sample sizes.

6.3 A real-world application

Finally, to illustrate the MPL for a high-dimensional real application, we consider a dataset of 1,000 aligned whole-genome DNA sequences of *Mycobacterium tuberculosis* (Casali et al., 2014). A Markov network can be used to reveal direct associations between variation over genome positions that may be relatively distant from each other. This purpose is similar

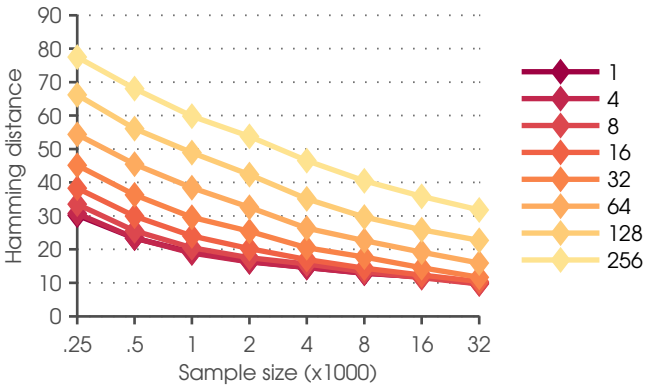


Figure 8: The MPL method applied on the Alarm network under different values of N. The average Hamming distance between the identified and true moral graph is plotted against the sample size.

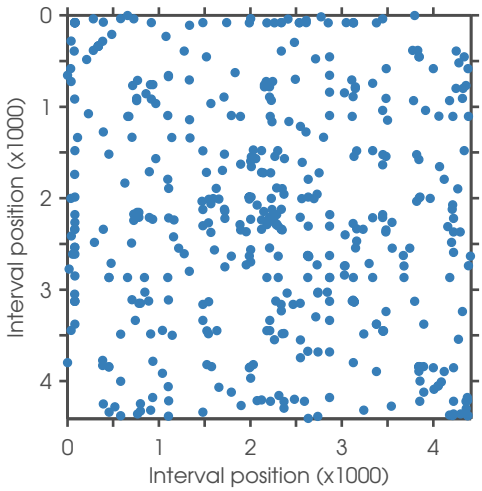


Figure 9: Association pattern identified by the MPL method for 1kb genome intervals in *M. tuberculosis*. A blue point indicates found linkage between positions that reside within the intervals located in the genome according to the values on the horizontal and vertical axes (in thousands of nucleotides).

to the use of Markov networks for finding direct dependencies among amino acid sequence positions that relate to the underlying crystal structure (Ekeberg et al., 2013).

The original 5Mb multiple sequence alignment for *M. tuberculosis* had approximately 27,000 variable positions, out of which we chose 589 that displayed sufficient variability determined by the threshold that the most frequent DNA base in a variable position was not allowed to represent more than 90% of the total number of observations. Similar to the amino acid dependency modeling with Markov networks, associations between genome positions that are close neighbors in the sequences are trivial and uninteresting for the biological purposes. We applied the MPL method on the 589 variables under the sparsity

promoting prior

$$p(G) \sim \prod_{j=1}^{d} 2^{-q_j(r_j-1)}.$$

in order to filter out the strongest dependencies.

The variable positions included in the analysis were separated maximally by almost 2.5 million bases since the bacterial genome is circular. To enable an efficient illustration of the results, the genome alignment was first split into non-overlapping intervals of 1000 (1kb) successive positions. Then, an adjacency matrix was created for the pairs of genome intervals such that a pair of intervals was determined adjacent if the identified graph contained an edge between any variable positions in the two intervals, respectively. The resulting adjacency structure is illustrated in Figure 9, which succinctly demonstrates the long-distance linkage of genome variation in this bacterium, as expected on the basis of its very low recombination rate.

7. Conclusions

In this work we have introduced a novel approach for learning the structure of a Markov network without imposing the restriction of chordality. Our marginal pseudo-likelihood scoring method is proven to be consistent and can be considered a small sample analytical version of the information theoretic PIC criterion (Csiszár and Talata, 2006). We have designed two algorithms for finding the MPL-optimal solution after an initial prescan for plausible edges. For moderately sized candidate sets of Markov blankets, we have shown that it is possible to obtain an exact solution to the restricted global optimization problem using pseudo-boolean optimization. As a fast alternative to the exact method, we considered a greedy hill climbing approach, which gave near optimal performance for reasonable sample sizes. The straightforward possibility of parallel use of the MPL makes it a viable candidate for high-dimensional knowledge discovery.

In comparison with the other methods for structure learning of Markov networks, our MPL method was overall superior, and only slightly inferior to alternatives under fairly extreme "large d, small n"-settings, or when the underlying network contained hub nodes. Moreover, it should also be kept in mind that we chose the tuning parameter values for the CMI and L1LR method by optimizing their performance against the known underlying structures, to reduce the computational burden of the experiments. In a real data analysis scenario, it would be necessary to tune these methods, using for example cross-validation, which would plausibly have a negative effect on their performance. In this sense, the comparison was extremely fair for the alternative methods, since no parameter choice was made for the MPL by the resulting performance. In terms of execution time, the MPL method is not necessarily as fast as the other methods, however, the runtimes of the CMI and L1LR method are here reported under a best case scenario. If the value of the penalty parameter would be determined experimentally, the computation times would easily exceed those reported for the MPL method.

The main drawback of the MPL compared to the true ML is that the former in a sense over-specifies the dependence structure. As a result, the MPL is less data efficient than the ML, especially when the true network contains hub nodes. On the other hand, calculation

of the true ML remains still intractable for non-chordal Markov networks. Therefore, the attractive properties of the MPL, combined with its solid performance in our experiments, suggest that our approach has considerable potential for both applications and further theoretical development.

Appendix A.

In this appendix we prove Theorem 1 from Section 4:

Theorem *Let $G^* \in \mathcal{G}$ be the true graph structure, of a Markov network over $(X_1, \ldots, X_d)$, with the corresponding Markov blankets $mb(G^*) = \{mb^*(1), \ldots, mb^*(d)\}$. Let $\theta_{G^*} \in \Theta_{G^*}$ define the corresponding joint distribution to which G^* is faithful and from which a sample $\mathbf{x}$ of size n is obtained. The local MPL estimator*

$$\widehat{mb}(j) = \underset{mb(j) \subseteq V \setminus j}{\arg\max}\ p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}) \tag{21}$$

is consistent in the sense that $\widehat{mb}(j) = mb^*(j)$ eventually almost surely as $n \to \infty$ for $j = 1, \ldots, d$. Consequently, the global MPL estimator

$$\hat{G} = \underset{G \in \mathcal{G}}{\arg\max}\ \hat{p}(\mathbf{x} \mid G) \tag{22}$$

is consistent in the sense that $\hat{G} = G^*$ eventually almost surely as $n \to \infty$.

Proof The proof is based on an asymptotic comparison of the local log-MPL

$$\begin{aligned} \log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}) = &\sum_{l=1}^{q_j} [\log \Gamma(\alpha_{jl}) - \log \Gamma(n_{jl} + \alpha_{jl}) \\ &+ \sum_{i=1}^{r_j} (\log \Gamma(n_{ijl} + \alpha_{ijl}) - \log \Gamma(\alpha_{ijl}))] \end{aligned} \tag{23}$$

and the PIC criterion (Csiszár and Talata, 2006) which is defined by

$$\mathrm{PIC}(mb(j); \mathbf{x}) = -\sum_{l=1}^{q_j} \sum_{i=1}^{r_j} n_{ijl} \log \frac{n_{ijl}}{n_{jl}} + q_j \log n$$

according to our notation. The PIC estimator is then defined as the Markov blanket (or neighborhood) that minimizes the above score. Csiszár and Talata (2006) conclude that the PIC criterion is consistent (Theorem 2.1). Their proof is based on two key propositions (Proposition 4.1 and 5.1) which together rule out the possibility of overestimation as well as underestimation. They also note that their results remains valid if the penalty term is multiplied by any constant $c > 0$. Thus, any estimator asymptotically equivalent to the PIC estimator is also consistent.

We proceed by investigating the asymptotic behavior of the local log-MPL when $n \to \infty$. We re-arrange the terms in (23) and omit the constant term introducing an $O(1)$ error:

$$\begin{aligned}
&\sum_{l=1}^{q_j}[\log\Gamma(\alpha_{jl}) - \log\Gamma(n_{jl}+\alpha_{jl}) + \sum_{i=1}^{r_j}(\log\Gamma(n_{ijl}+\alpha_{ijl}) - \log\Gamma(\alpha_{ijl}))] \\
&= \sum_{l=1}^{q_j}[-\log\Gamma(n_{jl}+\alpha_{jl}) + \sum_{i=1}^{r_j}\log\Gamma(n_{ijl}+\alpha_{ijl})] + \sum_{l=1}^{q_j}[\log\Gamma(\alpha_{jl}) - \sum_{i=1}^{r_j}\log\Gamma(\alpha_{ijl})] \\
&= \sum_{l=1}^{q_j}[-\log\Gamma(n_{jl}+\alpha_{jl}) + \sum_{i=1}^{r_j}\log\Gamma(n_{ijl}+\alpha_{ijl})] + O(1)
\end{aligned}$$

We let $n \to \infty$ and apply Stirling's asymptotic formula

$$\log\Gamma(n) \to (n - \frac{1}{2})\log n - n + O(1)$$

on the remaining terms:

$$\begin{aligned}
&\sum_{l=1}^{q_j}[-\log\Gamma(n_{jl}+\alpha_{jl}) + \sum_{i=1}^{r_j}\log\Gamma(n_{ijl}+\alpha_{ijl})] + O(1) \\
&\to \sum_{l=1}^{q_j}[-(n_{jl}+\alpha_{jl}-\frac{1}{2})\log(n_{jl}+\alpha_{jl}) + (n_{jl}+\alpha_{jl}) \\
&+ \sum_{i=1}^{r_j}(n_{ijl}+\alpha_{ijl}-\frac{1}{2})\log(n_{ijl}+\alpha_{ijl}) - (n_{ijl}+\alpha_{ijl})] + O(1) \\
&= \sum_{l=1}^{q_j}\sum_{i=1}^{r_j} n_{ijl}\log\frac{n_{ijl}+\alpha_{ijl}}{n_{jl}+\alpha_{jl}} + \sum_{l=1}^{q_j}\sum_{i=1}^{r_j}\alpha_{ijl}\log\frac{n_{ijl}+\alpha_{ijl}}{n_{jl}+\alpha_{jl}} \\
&+ \sum_{l=1}^{q_j}[\frac{1}{2}\log(n_{jl}+\alpha_{jl}) - \sum_{i=1}^{r_j}\frac{1}{2}\log(n_{ijl}+\alpha_{ijl})] + O(1)
\end{aligned}$$

The second step is allowed since $n_{jl} = \sum_{i=1}^{r_j} n_{ijl}$ and $\alpha_{jl} = \sum_{i=1}^{r_j}\alpha_{ijl}$. As $n \to \infty$ we have that

$$\frac{n_{ijl}+\alpha_{ijl}}{n_{jl}+\alpha_{jl}} = \frac{n_{ijl}(1+\frac{\alpha_{ijl}}{n_{ijl}})}{n_{jl}(1+\frac{\alpha_{jl}}{n_{jl}})} \to \frac{n_{ijl}}{n_{jl}}$$

Since n_{ijl}/n_{jl} is the maximum likelihood estimate of the parameter θ_{ijl} we further know that

$$\sum_{l=1}^{q_j}\sum_{i=1}^{r_j}\alpha_{ijl}\log\frac{n_{ijl}+\alpha_{ijl}}{n_{jl}+\alpha_{jl}} = O(1)$$

Finally, the remaining terms can be rewritten as

$$\begin{aligned}
&\sum_{l=1}^{q_j}[\frac{1}{2}\log(n_{jl}+\alpha_{jl}) - \sum_{i=1}^{r_j}\frac{1}{2}\log(n_{ijl}+\alpha_{ijl})] \\
&= \frac{1}{2}\sum_{l=1}^{q_j}[\log\frac{n_{jl}+\alpha_{jl}}{n} - \log n - \sum_{i=1}^{r_j}(\log\frac{n_{ijl}+\alpha_{ijl}}{n} - \log n)] \\
&= \frac{1}{2}\sum_{l=1}^{q_j}[-\log n + \sum_{i=1}^{r_j}\log n] + \frac{1}{2}\sum_{l=1}^{q_j}[\log\frac{n_{jl}+\alpha_{jl}}{n} - \sum_{i=1}^{r_j}\log\frac{n_{ijl}+\alpha_{ijl}}{n}] \\
&= \frac{1}{2}\sum_{l=1}^{q_j}[-\log n + \sum_{i=1}^{r_j}\log n] + O(1) \\
&= \frac{1}{2}(-q_j\log n + q_j r_j\log n) + O(1) \\
&= \frac{q_j(r_j-1)}{2}\log n + O(1)
\end{aligned}$$

Piecing everything together,

$$\log p(\mathbf{x}_j \mid \mathbf{x}_{mb(j)}) \to \sum_{l=1}^{q_j}\sum_{i=1}^{r_j} n_{ijl}\log\frac{n_{ijl}}{n_{jl}} - \frac{(r_j-1)q_j}{2}\log n + O(1).$$

as $n \to \infty$. Since the $O(1)$ term does not grow with n, the local log-MPL is asymptotically equivalent to

$$\sum_{l=1}^{q_j}\sum_{i=1}^{r_j} n_{ijl}\log\frac{n_{ijl}}{n_{jl}} - c_j \cdot q_j \log n.$$

where $c_j = (r_j - 1)/2$ is a variable specific constant. Consequently, the local MPL estimator is asymptotically equivalent to minimizing

$$-\sum_{l=1}^{q_j}\sum_{i=1}^{r_j} n_{ijl}\log\frac{n_{ijl}}{n_{jl}} + c_j \cdot q_j \log n$$

which is equivalent to the consistent PIC estimator up to a constant factor on the penalty term. Hence the local MPL estimator (21) is consistent.

Since the local MPL estimator is consistent, the true collection of Markov blankets is eventually identified when $n \to \infty$. A set of Markov blankets uniquely specifies the structure of a Markov network. Since the true model structure satisfies the structural properties of a Markov network, that is $i \in mb(j)$ if $j \in mb(i)$, the global MPL estimator (22) is also consistent. ■

Appendix B.

This appendix contains supplementary material of Section 6 in form of numerical results.

d	n	Number of solved instances	PBO			HC		
			Encoding/Solving time (s)	log-MPL	TP/FP	Time (s)	log-MPL	TP/FP
64	250	100	0.31/1.24	-8700.20	**34.16/6.23**	0.11	-8701.07	33.79/6.11
	500	100	0.31/1.13	-16974.31	**41.12/2.97**	0.13	-16976.75	40.77/2.78
	1000	100	0.26/0.67	-33310.55	**49.08/1.06**	0.16	-33317.19	48.55/1.08
	2000	100	0.46/0.77	-65613.39	**55.75/0.46**	0.25	-65614.25	55.62/0.45
	4000	100	0.70/0.76	-132021.36	61.84/0.18	0.40	-132021.37	**61.85/0.17**
	8000	100	1.47/0.94	-262370.09	**65.33/0.13**	0.74	-262370.12	**65.33/0.13**
	16000	100	1.57/0.49	-519632.76	**65.51/0.10**	1.89	-519632.76	**65.51/0.10**
	32000	100	4.04/0.76	-1015034.17	68.51/0.03	3.73	-1015034.34	**68.52/0.03**
128	250	91	4.42/43.63	-17017.35	62.13/25.73	0.22	-17021.41	**61.62/23.91**
	500	100	1.22/7.19	-33552.92	**79.52/10.43**	0.23	-33561.81	78.45/10.22
	1000	100	1.39/4.33	-66074.15	**97.25/4.42**	0.29	-66081.21	96.46/4.46
	2000	100	0.92/2.05	-131739.79	**112.87/2.26**	0.47	-131742.17	112.68/2.22
	4000	99	1.31/1.83	-258402.88	**122.91/0.94**	0.76	-258403.11	122.88/0.92
	8000	100	2.18/1.71	-514098.80	**127.50/0.46**	1.39	-514098.95	127.45/0.46
	16000	100	3.56/1.78	-1030570.26	**133.66/0.26**	4.00	-1030570.26	**133.66/0.26**
	32000	100	8.77/2.09	-2086638.66	**137.16/0.09**	7.81	-2086639.27	**137.16/0.09**
256	250	10	12.79/235.78	-34505.12	120.70/64.00	0.48	-34521.43	**120.30/60.50**
	500	55	13.51/118.93	-67470.66	**161.13/31.95**	0.49	-67482.09	159.60/30.84
	1000	90	6.26/28.85	-132908.96	**199.94/13.91**	0.63	-132927.70	198.66/13.52
	2000	96	6.33/17.04	-261032.09	**222.43/7.21**	0.92	-261044.24	221.75/7.02
	4000	100	4.65/7.33	-518442.23	**245.28/3.04**	1.56	-518446.10	245.17/2.99
	8000	100	6.25/6.56	-1031306.39	**257.04/1.63**	2.82	-1031306.65	**257.02/1.61**
	16000	100	8.53/4.22	-2076927.13	**267.76/0.47**	8.13	-2076927.30	267.70/0.47
	32000	100	17.04/3.75	-4075373.11	**275.12/0.32**	15.25	-4075375.01	275.06/0.32
512	250	0	-	-	-	-	-	-
	500	0	-	-	-	-	-	-
	1000	19	19.78/179.25	-267311.61	**379.37/42.00**	1.37	-267355.80	375.58/41.79
	2000	44	9.24/44.10	-522454.75	**447.18/19.23**	1.99	-522464.46	446.55/18.84
	4000	77	14.93/39.80	-1043048.89	**485.06/9.47**	3.20	-1043055.03	484.78/9.38
	8000	97	15.08/19.84	-2075735.29	513.11/4.18	5.84	-2075735.89	**513.11/4.14**
	16000	97	18.58/10.94	-4166121.62	**532.99/2.19**	16.82	-4166134.18	532.82/2.19
	32000	100	34.25/8.71	-8296145.22	**553.44/1.06**	31.54	-8296146.02	553.38/1.06

Table 3: Results from the comparison of the PBO and HC method in Section 6.1. (**bold font** = lowest Hamming distance)

d	n	MPL	PIC	CMI		L1LR	
				AND	OR	AND	OR
64	250	**33.79/6.11**	23.77/0.34	22.74/1.29	23.22/2.36	24.11/1.74	26.96/2.26
	500	**40.77/2.78**	29.43/0.09	29.03/0.45	24.44/1.58	32.31/3.23	34.38/3.31
	1000	**48.55/1.08**	39.09/0.07	41.65/5.87	37.57/3.20	39.86/1.69	44.54/2.86
	2000	**55.62/0.45**	46.81/0.02	49.18/1.30	37.37/0.12	47.41/1.99	50.50/3.65
	4000	**61.85/0.17**	55.90/0.01	57.16/0.49	58.98/4.66	55.66/1.19	59.62/2.52
	8000	**65.33/0.13**	60.92/0.01	62.70/1.59	57.32/2.20	62.51/1.41	65.20/2.96
	16000	**65.51/0.10**	61.84/0.00	61.42/0.58	59.29/1.51	60.23/0.72	63.52/2.22
	32000	**68.52/0.03**	65.47/0.00	68.45/0.03	58.61/0.87	66.13/0.84	66.60/1.92
128	250	61.09/24.04	**44.28/1.85**	38.10/0.54	37.33/2.82	40.48/3.06	46.38/6.39
	500	**78.45/10.22**	58.32/0.51	57.15/0.73	46.68/1.14	59.79/5.08	67.32/11.16
	1000	**96.46/4.46**	75.49/0.20	81.98/11.04	69.99/10.80	73.98/2.51	82.76/7.35
	2000	**112.68/2.22**	97.49/0.04	97.53/3.25	73.20/0.64	95.28/4.43	101.51/11.33
	4000	**122.88/0.93**	111.89/0.06	108.58/1.87	114.37/9.46	105.43/2.44	111.84/6.69
	8000	**127.45/0.46**	118.91/0.00	121.44/2.99	117.22/8.50	116.72/3.78	117.89/5.81
	16000	**133.66/0.26**	126.10/0.01	125.24/0.25	119.42/3.72	124.84/2.22	127.07/4.28
	32000	**137.16/0.09**	130.81/0.00	136.62/0.18	122.34/1.35	131.61/2.84	130.49/2.15
256	250	118.30/66.51	**87.70/6.81**	75.94/0.79	68.19/1.55	81.55/12.82	90.90/21.09
	500	**158.28/32.72**	119.64/1.98	117.51/2.05	93.90/1.36	123.63/20.07	127.14/28.66
	1000	**198.98/13.76**	157.33/0.53	168.82/22.57	146.36/32.54	150.65/9.98	163.52/19.68
	2000	**221.73/7.08**	188.35/0.22	191.71/5.95	145.50/2.03	183.54/15.62	175.56/10.07
	4000	**245.17/2.99**	222.17/0.05	223.00/3.08	225.08/28.10	210.33/9.00	210.06/7.13
	8000	**257.02/1.61**	239.52/0.03	242.47/5.14	225.79/17.45	236.47/13.77	227.10/4.39
	16000	**267.70/0.47**	255.05/0.01	253.66/0.27	237.94/7.68	253.64/8.56	248.94/6.57
	32000	**275.06/0.32**	264.72/0.00	271.74/0.07	243.56/2.18	262.54/9.02	253.13/5.90
512	250	238.83/168.39	**182.13/22.64**	159.25/2.77	145.26/5.19	160.15/37.21	134.20/4.10
	500	303.82/85.51	**235.79/7.33**	228.66/4.57	189.02/7.22	206.20/40.99	187.37/2.49
	1000	**384.26/40.57**	306.16/2.05	333.11/35.84	282.97/74.35	298.57/38.14	276.56/6.36
	2000	**442.16/20.10**	377.77/0.63	374.03/8.48	287.87/3.36	345.15/42.16	325.34/3.19
	4000	**484.44/9.65**	439.55/0.17	432.42/4.06	432.73/60.98	417.76/30.00	406.06/9.57
	8000	**513.00/4.17**	476.57/0.04	485.44/7.56	442.44/33.06	433.84/12.07	434.82/14.64
	16000	**532.61/2.25**	504.79/0.04	502.86/0.25	470.29/23.79	489.49/15.26	447.91/15.35
	32000	**553.38/1.06**	529.72/0.02	545.69/0.12	488.06/6.44	505.80/1.25	457.72/14.94

Table 4: True and false positives (TP/FP) for the methods used in Section 6.1. (**bold font** = lowest Hamming distance)

d	n	Markov blanket discovery (total/node max)						Hill climbing	
		MPL	PIC	CMI		L1LR		MPL	PIC
				AND	OR	AND	OR		
64	250	1.83/0.11	1.25/0.06	1.17/0.11	0.52/0.05	0.96/0.05	0.97/0.05	0.11	0.07
	500	2.18/0.13	1.61/0.08	1.76/0.25	0.61/0.06	1.23/0.06	1.26/0.06	0.13	0.09
	1000	2.93/0.15	2.42/0.11	7.93/0.52	1.08/0.16	1.66/0.08	1.65/0.08	0.16	0.12
	2000	4.83/0.25	4.19/0.21	10.08/1.23	1.42/0.10	3.51/0.16	3.48/0.16	0.25	0.20
	4000	8.40/0.39	7.70/0.33	18.26/2.60	4.56/0.91	9.97/0.46	9.86/0.46	0.40	0.35
	8000	15.80/0.85	14.81/0.64	55.00/7.12	7.70/1.19	24.81/1.02	23.76/1.00	0.74	0.66
	16000	45.14/2.20	42.34/1.82	65.86/13.39	18.19/2.33	69.83/3.05	67.39/2.99	1.89	1.38
	32000	84.43/4.96	78.33/3.81	147.63/40.20	43.89/4.52	164.17/6.71	148.07/6.09	3.73	2.81
128	250	8.45/0.24	4.78/0.10	2.97/0.24	1.82/0.09	2.84/0.06	2.84/0.06	0.22	0.10
	500	9.02/0.24	6.39/0.16	8.03/0.54	2.31/0.12	3.75/0.08	3.74/0.08	0.23	0.14
	1000	12.14/0.32	9.45/0.24	42.90/1.20	4.57/0.58	6.57/0.14	6.57/0.14	0.29	0.20
	2000	20.30/0.49	17.57/0.46	47.29/2.39	5.75/0.28	18.63/0.42	18.43/0.41	0.47	0.38
	4000	33.77/0.81	30.61/0.68	78.82/5.24	17.21/2.29	37.88/0.80	36.67/0.78	0.76	0.66
	8000	62.61/1.74	58.88/1.43	209.36/13.22	34.73/5.71	117.10/2.27	105.14/2.10	1.39	1.21
	16000	186.96/5.18	174.55/3.96	160.95/26.09	74.74/8.73	337.29/7.65	294.69/6.64	4.00	2.86
	32000	343.98/11.83	323.12/9.15	510.09/82.21	185.69/11.66	829.72/19.47	668.22/15.11	7.81	5.60
256	250	40.96/0.67	19.63/0.22	13.21/0.55	6.77/0.16	8.61/0.09	8.66/0.09	0.54	0.17
	500	41.38/0.60	26.24/0.32	41.37/1.11	9.19/0.23	15.18/0.17	15.56/0.17	0.50	0.24
	1000	52.61/0.72	39.49/0.58	218.80/2.40	20.05/1.45	41.89/0.44	41.60/0.44	0.63	0.38
	2000	80.98/1.10	67.61/0.90	222.70/4.80	22.87/0.95	99.96/1.03	88.52/0.94	0.92	0.67
	4000	138.11/2.08	125.06/1.39	342.81/10.50	69.60/6.18	218.26/2.41	186.89/2.02	1.56	1.29
	8000	256.81/4.36	238.05/3.13	959.13/27.50	125.43/14.78	573.22/7.45	406.35/5.13	2.82	2.44
	16000	755.30/11.57	710.80/9.12	651.17/57.73	284.78/26.34	1536.93/21.77	1110.54/14.63	8.13	5.74
	32000	1404.29/27.84	1299.78/19.75	1996.34/171.50	719.29/31.50	3150.53/41.52	1948.53/26.06	15.25	10.97
512	250	208.35/2.16	83.19/0.48	67.56/1.17	28.05/0.51	39.11/0.22	43.20/0.23	1.59	0.37
	500	183.72/1.33	105.99/0.65	193.05/2.27	37.88/0.99	90.13/0.47	93.68/0.46	1.24	0.49
	1000	220.51/1.53	156.32/1.25	1001.81/4.73	81.83/3.12	192.60/1.03	182.68/0.92	1.41	0.76
	2000	334.71/2.41	272.74/1.80	1039.68/9.91	89.86/2.44	443.78/2.78	369.68/1.96	2.01	1.37
	4000	557.18/4.62	494.41/2.82	1445.58/21.49	276.06/14.89	1131.22/7.96	862.90/5.38	3.20	2.54
	8000	1031.36/9.70	957.13/6.93	4374.04/55.34	472.23/31.79	2404.33/17.21	1925.95/13.67	5.85	4.95
	16000	3058.00/29.74	2842.66/19.69	2261.52/111.66	1057.63/71.35	7002.39/45.74	3614.59/27.36	16.81	11.28
	32000	5697.65/64.94	5337.01/44.28	8494.37/355.76	2782.49/98.06	16049.32/92.76	8041.91/59.57	31.54	22.68

Table 5: Execution times (in seconds) for the methods used in Section 6.1.

Network	n	MPL	PIC	CMI	
				AND	OR
Alarm	250	**38.48/3.70**	17.87/1.87	19.55/2.53	30.73/5.71
	500	**43.05/1.39**	19.96/2.04	22.48/0.93	33.51/3.00
	1000	**46.22/0.39**	25.62/2.02	28.15/2.73	37.60/3.92
	2000	**49.02/0.37**	31.59/1.40	30.37/0.06	39.39/2.38
	4000	**50.65/0.25**	34.94/0.99	34.44/2.00	42.22/2.63
	8000	**52.40/0.35**	39.00/0.34	39.76/0.50	47.28/1.95
	16000	**53.98/0.74**	42.77/0.21	43.78/0.27	49.57/2.02
	32000	**55.97/0.98**	44.00/1.02	45.45/0.57	53.98/1.58
Insurance	250	**21.75/1.95**	14.37/0.69	15.67/3.02	18.31/2.75
	500	**25.26/1.44**	16.82/0.93	18.75/1.41	30.26/10.90
	1000	**28.95/1.02**	20.21/1.11	21.06/1.54	32.98/8.22
	2000	**32.04/0.82**	22.12/0.97	22.97/2.81	31.95/4.61
	4000	**34.94/0.78**	26.03/0.93	27.62/2.91	33.32/2.56
	8000	**38.67/0.87**	28.63/0.92	31.91/2.40	37.09/4.52
	16000	**40.87/0.92**	31.02/0.88	32.52/1.05	37.85/1.03
	32000	**43.61/0.82**	34.00/0.99	40.63/2.32	36.99/0.99
Hailfinder	250	25.83/8.97	**21.52/3.34**	16.28/2.58	18.18/4.65
	500	30.83/9.25	**25.50/3.01**	16.92/3.54	20.14/2.88
	1000	36.08/10.31	30.90/2.19	25.28/6.85	**35.50/1.66**
	2000	38.60/10.64	35.45/2.65	23.16/3.54	**34.95/1.31**
	4000	43.68/9.54	38.69/3.20	28.05/0.23	**49.81/4.59**
	8000	52.09/9.42	43.05/2.81	34.66/0.02	**50.41/4.82**
	16000	55.71/8.69	47.57/2.78	35.75/0.17	**52.38/5.25**
	32000	**59.21/8.84**	50.68/3.89	37.01/0.00	52.84/4.89
Barley	250	**16.69/3.93**	13.52/2.39	3.25/0.67	4.90/1.73
	500	19.86/3.61	**20.64/2.99**	9.55/3.17	14.70/6.15
	1000	**27.28/3.18**	24.84/2.99	16.28/4.37	22.59/8.69
	2000	**32.67/3.45**	28.89/3.02	17.65/5.78	22.98/6.00
	4000	**39.68/3.25**	30.53/3.00	19.37/5.15	26.26/6.16
	8000	**44.03/3.39**	34.18/3.00	17.52/2.00	40.63/13.47
	16000	**49.49/3.03**	39.40/3.21	19.27/2.60	37.68/14.61
	32000	**54.95/3.09**	39.71/3.00	24.67/2.13	42.00/8.00

(a)

n	N	MPL	N	MPL
250	1	**38.48/3.70**	32	45.38/25.52
500		**43.05/1.39**		48.16/19.47
1000		46.22/0.39		50.31/14.92
2000		49.02/0.37		51.33/11.72
4000		**50.65/0.25**		53.04/8.53
8000		52.40/0.35		54.40/7.08
16000		53.98/0.74		55.78/5.30
32000		55.97/0.98		57.39/4.02
250	4	40.77/6.49	64	45.96/35.34
500		44.75/3.16		48.67/29.15
1000		**47.39/1.29**		50.67/24.12
2000		**49.48/0.75**		51.92/19.47
4000		**50.85/0.45**		53.95/15.30
8000		**53.01/0.83**		55.38/12.99
16000		54.20/0.98		56.28/10.42
32000		56.26/1.01		57.25/8.19
250	8	42.33/10.85	128	46.40/47.58
500		45.76/6.28		49.27/40.37
1000		48.16/3.65		50.72/34.63
2000		50.03/2.51		52.49/29.80
4000		51.30/1.95		54.63/24.74
8000		53.38/1.70		55.96/20.65
16000		**54.80/1.33**		56.58/17.48
32000		**56.46/1.11**		57.20/14.91
250	16	44.11/17.37	256	47.00/59.48
500		47.08/12.02		49.79/52.86
1000		49.24/8.28		51.69/46.52
2000		50.69/5.91		52.78/41.55
4000		52.12/4.10		54.71/36.27
8000		53.93/3.31		56.12/31.58
16000		55.47/2.78		56.76/27.60
32000		56.70/2.12		57.55/24.35

(b)

Table 6: True and false positives (TP/FP) for the simulations in Section 6.2 for (a) the methods used in the comparison, and (b) the MPL-HC applied on the Alarm network under different values of the equivalent sample size parameter N. (**bold font** = lowest Hamming distance)

Network	n	Markov blanket discovery (total/node max)				Hill climbing	
		MPL	PIC	CMI		MPL	PIC
				AND	OR		
Alarm	250	1.31/0.09	0.67/0.04	0.43/0.06	0.28/0.03	0.18	0.08
	500	1.71/0.12	0.94/0.04	0.53/0.08	0.33/0.02	0.23	0.10
	1000	2.42/0.19	1.38/0.06	1.15/0.27	0.50/0.05	0.33	0.13
	2000	3.96/0.35	2.32/0.13	1.34/0.38	0.75/0.06	0.54	0.21
	4000	7.21/0.73	4.08/0.22	4.18/1.08	1.53/0.19	0.98	0.35
	8000	14.92/1.69	8.28/0.43	9.25/3.47	3.28/0.22	2.09	0.74
	16000	37.84/4.52	23.56/1.51	23.52/8.39	7.58/0.50	5.32	1.98
	32000	78.17/10.79	47.98/3.31	59.65/21.41	22.09/1.53	11.48	3.99
Insurance	250	0.64/0.06	0.45/0.04	0.33/0.04	0.13/0.01	0.11	0.07
	500	0.89/0.08	0.62/0.04	0.46/0.06	0.29/0.05	0.13	0.09
	1000	1.37/0.12	0.93/0.05	0.75/0.13	0.40/0.08	0.21	0.13
	2000	2.36/0.20	1.61/0.14	1.51/0.28	0.50/0.10	0.40	0.20
	4000	4.36/0.58	2.99/0.25	3.52/0.78	0.83/0.15	0.73	0.43
	8000	8.94/1.27	5.85/0.47	6.82/1.57	2.13/0.67	1.64	0.79
	16000	25.27/3.26	15.28/1.62	12.66/3.24	3.85/0.59	4.43	1.94
	32000	57.61/11.37	35.42/4.91	55.46/15.08	9.45/1.16	10.69	4.86
Hailfinder	250	3.02/0.19	2.68/0.09	1.64/0.05	0.37/0.03	0.19	0.12
	500	4.13/0.22	3.69/0.13	2.00/0.10	0.43/0.03	0.31	0.16
	1000	5.94/0.30	5.19/0.19	4.61/0.20	0.66/0.03	0.65	0.24
	2000	8.98/0.41	8.98/0.44	4.84/0.43	0.89/0.04	1.44	0.38
	4000	15.02/0.75	15.71/0.96	5.77/0.79	2.34/0.12	3.71	0.64
	8000	32.41/2.16	28.81/1.63	18.35/1.92	4.74/0.36	13.24	1.19
	16000	82.42/6.04	68.38/5.11	39.27/4.99	10.72/0.95	35.46	2.84
	32000	179.16/22.53	152.45/17.70	47.21/9.80	26.04/1.75	124.36	6.20
Barley	250	3.64/0.28	2.80/0.15	0.28/0.03	0.22/0.02	0.13	0.10
	500	6.92/0.68	5.93/0.29	1.15/0.08	0.44/0.06	0.20	0.18
	1000	11.20/1.02	10.56/0.51	3.45/0.23	0.75/0.15	0.37	0.31
	2000	19.93/2.31	18.70/1.13	7.94/0.47	0.91/0.24	0.73	0.52
	4000	50.08/6.39	34.09/1.92	12.24/1.08	1.89/0.66	2.12	1.11
	8000	117.97/13.93	71.36/4.79	7.46/2.70	7.46/2.70	4.52	2.22
	16000	346.72/47.56	155.19/10.12	26.38/5.73	16.36/4.72	11.43	5.45
	32000	921.36/173.24	357.36/26.95	103.51/16.75	33.59/16.90	36.83	12.74

Table 7: Execution times (in seconds) for the methods used in Section 6.2.

d	n	CMI		L1LR	
		AND	OR	AND	OR
64	250	0.02500/0.05000	0.05000/0.10000	8/16	12/16
	500	0.02500/0.02500	0.02500/0.05000	12/16	16/24
	1000	0.01000/0.02500	0.02500/0.05000	16/24	24/32
	2000	0.00500/0.01000	0.01000/0.02500	24/32	24/48
	4000	0.00500/0.00750	0.00750/0.01000	32/48	48/64
	8000	0.00250/0.00500	0.00500/0.01000	48/64	64/96
	16000	0.00100/0.00250	0.00250/0.01000	64/128	96/128
	32000	0.00075/0.00100	0.00100/0.00750	128/192	128/256
128	250	0.02500/0.05000	0.05000/0.07500	12/12	12/16
	500	0.02500/0.02500	0.05000/0.07500	16/16	16/24
	1000	0.01000/0.02500	0.02500/0.05000	24/24	24/32
	2000	0.00500/0.01000	0.02500/0.02500	32/32	32/48
	4000	0.00500/0.00750	0.00750/0.01000	48/48	48/64
	8000	0.00250/0.00500	0.00500/0.01000	64/96	64/96
	16000	0.00100/0.00250	0.00250/0.01000	96/128	96/128
	32000	0.00075/0.00100	0.00250/0.00750	128/192	128/256
256	250	0.05000/0.05000	0.07500/0.07500	12/16	12/16
	500	0.02500/0.02500	0.05000/0.05000	16/16	16/24
	1000	0.01000/0.02500	0.02500/0.05000	24/24	24/32
	2000	0.00500/0.01000	0.02500/0.02500	32/32	32/48
	4000	0.00500/0.00750	0.00750/0.01000	48/48	48/64
	8000	0.00250/0.00500	0.00500/0.01000	64/96	64/128
	16000	0.00100/0.00250	0.00500/0.01000	96/128	96/192
	32000	0.00075/0.00100	0.00250/0.00750	128/192	192/512
512	250	0.05000/0.05000	0.07500/0.07500	12/16	12/16
	500	0.02500/0.02500	0.05000/0.05000	16/24	24/24
	1000	0.01000/0.01000	0.02500/0.05000	24/32	24/32
	2000	0.00500/0.01000	0.02500/0.02500	32/48	48/64
	4000	0.00500/0.00500	0.01000/0.01000	48/64	64/96
	8000	0.00250/0.00250	0.00750/0.01000	64/96	96/128
	16000	0.00100/0.00250	0.00500/0.01000	96/128	128/256
	32000	0.00100/0.00100	0.00500/0.00750	192/192	256/512

(a)

Network	n	CMI	
		AND	OR
Alarm	250	0.05000/0.10000	0.07500/0.25000
	500	0.02500/0.07500	0.05000/0.10000
	1000	0.02500/0.05000	0.05000/0.10000
	2000	0.02500/0.02500	0.02500/0.07500
	4000	0.01000/0.02500	0.02500/0.05000
	8000	0.00750/0.01000	0.02500/0.02500
	16000	0.00500/0.01000	0.01000/0.02500
	32000	0.00250/0.00750	0.00750/0.01000
Insurance	250	0.05000/0.25000	0.07500/0.25000
	500	0.05000/0.10000	0.07500/0.25000
	1000	0.05000/0.10000	0.05000/0.10000
	2000	0.02500/0.07500	0.05000/0.10000
	4000	0.02500/0.05000	0.05000/0.10000
	8000	0.01000/0.02500	0.05000/0.07500
	16000	0.00750/0.02500	0.05000/0.05000
	32000	0.00750/0.01000	0.05000/0.05000
Hailfinder	250	0.05000/0.25000	0.50000/0.75000
	500	0.02500/0.25000	0.25000/0.50000
	1000	0.02500/0.10000	0.25000/0.25000
	2000	0.05000/0.10000	0.25000/0.25000
	4000	0.07500/0.10000	0.10000/0.10000
	8000	0.05000/0.05000	0.07500/0.10000
	16000	0.02500/0.05000	0.07500/0.10000
	32000	0.05000/0.05000	0.05000/0.10000
Barley	250	0.10000/2.00000	1.25000/2.00000
	500	0.01000/1.25000	1.00000/1.25000
	1000	0.00500/0.50000	0.75000/1.00000
	2000	0.00500/0.25000	0.75000/0.75000
	4000	0.00500/0.50000	0.50000/0.50000
	8000	0.25000/0.25000	0.25000/0.25000
	16000	0.10000/0.25000	0.25000/0.25000
	32000	0.07500/0.10000	0.25000/0.25000

(b)

Table 8: Minimum and maximum tuning parameter values (min/max) picked by (a) the CMI and L1LR method in Section 6.1, and (b) the CMI method in Section 6.2.

References

T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1:1–41, 2009.

H. Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19:716–723, 1974.

A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky. High-dimensional structure estimation in Ising models: Local separation criterion. *The Annals of Statistics*, 40: 1346–1375, 2012.

E. Aurell and M. Ekeberg. Inverse Ising inference using all the data. *Physical Review Letters*, 108:090201, 2012.

M. Bartlett and J. Cussens. Advances in Bayesian network learning using integer programming. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI 2013)*, pages 182–191. AUAI Press, 2013.

J. Berg, M. Järvisalo, and B. Malone. Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *Proceedings of the 17th Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, 2014.

T. Berthold, S. Heinz, and M. E. Pfetsch. Solving pseudo-boolean problems with SCIP. ZIB-report, Zuse Institute Berlin, 2009.

J. E. Besag. Nearest-neighbour systems and the auto-logistic model for binary data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34:75–83, 1972.

E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:155–225, 2002.

F. Bromberg, D. Margaritis, and V. Honavar. Efficient Markov network structure discovery using independence tests. *Journal of Artificial Intelligence Research*, 35:449–485, 2009.

W. Buntine. Theory refinement on Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann, 1991.

N. Casali, V. Nikolayevskyy, Y. Balabanova, S. R. Harris, O. Ignatyeva, I. Kontsevaya, J. Corander, J. Bryant, J. Parkhill, S. Nejentsev, R. D. Horstmann, T. Brown, and F. Drobniewski. Evolution and transmission of drug resistant tuberculosis in a population: Insights from a 1000 genome study. *Nature Genetics*, 46:279–286, 2014.

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

J. Corander, M. Ekdahl, and T. Koski. Parallel interacting MCMC for learning of topologies of graphical models. *Data Mining and Knowledge Discovery*, 17:431–456, 2008.

J. Corander, T. Janhunen, J. Rintanen, H. Nyman, and J. Pensar. Learning chordal Markov networks by constraint satisfaction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1349–1357, 2013.

I. Csiszár and Z. Talata. Consistent estimation of the basic neighborhood of Markov random fields. *Annals of Statistics*, 34:123–145, 2006.

J. Cussens. Bayesian network learning by compiling to weighted MAX-SAT. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 105–112, 2008.

M. Ekeberg, C. Lövkvist, Y. Lan, M. Weigt, and E. Aurell. Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models. *Physical Review E*, 87:012707, 2013.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In E. Horvitz and F. V. Jensen, editors, *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pages 252–262. Morgan Kaufmann, 1996.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2001.

H. Höfling and R. Tibshirani. Estimation of sparse binary pairwise Markov networks using pseudo-likelihoods. *Journal of Machine Learning Research*, 10:883–906, 2009.

C. Ji and L. Seymour. A consistent model selection procedure for Markov random fields based on penalized pseudolikelihood. *Annals of Applied Probability*, 6:423–443, 1996.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, 1996.

S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using ℓ_1-regularization. In *Advances in Neural Information Processing Systems*, 2006.

D. Lowd and J. Davis. Improving Markov network structure learning using decision trees. *Journal of Machine Learning Research*, 15:501–532, 2014.

P. Parviainen, H.S. Farahani, and J. Lagergren. Learning bounded tree-width Bayesian networks using integer linear programming. In *Proceedings of the 17th Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, 2014.

H. Peng. Mutual information computation package, 2007, Accessed 2013-10-14. URL `http://www.mathworks.com/matlabcentral/fileexchange/14888-mutual-\information-computation`.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393, 1997.

P. Ravikumar, M. J. Wainwright, and J. D. Lafferty. High-dimensional Ising model selection using ℓ_1-regularized logistic regression. *Annals of Statistics*, 38:1287–1319, 2010.

M. Schmidt. Matlab code by Mark Schmidt (2005-2013), 2013, Accessed 2013-10-14. URL `http://www.di.ens.fr/$\sim$mschmidt/Software/code.html`.

M. Schmidt and K. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 2010.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

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

T. Silander, P. Kontkanen, and P. Myllymäki. On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In R. Parr and L. van der Gaag, editors, *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 360–367. AUAI Press, 2007.

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

I. Tsamardinos, C. Aliferis, A. Statnikov, and E. Statnikov. Algorithms for large scale Markov blanket discovery. In *The 16th International FLAIRS Conference*, pages 376–380. AAAI Press, 2003.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78, 2006.