MCMC Learning of Bayesian Network Models by Markov Blanket Decomposition

Carsten Riggelsen

Institute of Information & Computing Sciences,
Utrecht University, P.O. Box 80.098, 3508TB, Utrecht, The Netherlands
carsten@cs.uu.nl

Abstract. We propose a Bayesian method for learning Bayesian network models using Markov chain Monte Carlo (MCMC). In contrast to most existing MCMC approaches that define components in term of single edges, our approach is to decompose a Bayesian network model in larger *dependence* components defined by Markov blankets. The idea is based on the fact that MCMC performs significantly better when choosing the right decomposition, and that edges in the Markov blanket of the vertices form a natural dependence relationship. Using the ALARM and Insurance networks, we show that this decomposition allows MCMC to mix more rapidly, and is less prone to getting stuck in local maxima compared to the single edge approach.

1 Introduction

Bayesian networks is a convenient framework for reasoning and manipulating beliefs about the real world. It occupies a prominent position in decision support environments where it is used for diagnostic and prediction purposes. Also, in the context of data mining especially the graphical structure (model) of a Bayesian network is an appealing formalism for visualising the relationships between domain variables. This paper is concerned with the latter of the two.

We approach model learning from a Bayesian point of view, and apply a method that is based on the marginal likelihood scoring criterion. The reason for being Bayesian is first of all related to the relatively small amount of data that we often have at our disposal in practice. When data is scarce, there may be several models that are structurally quite different from each other, yet are almost equally likely given the data. In other words, the data supports several models that differ widely, yet from a scoring perspective are very close. Model selection methods on the other hand will return "the best" model, but give no clue as to how and in what respect models differ that score almost equally well.

For large data sets—where "large" of course is related to the number of variables of our domain—the best model is much more likely than any other model, and model selection may be adequate.

The usual obstacle with the Bayesian statistical approach is the analytically intractable integrals, normalising constants and large mixtures encountered when performing the required computations. With MCMC, Bayesian computations are performed using stochastic simulation.

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 329–340, 2005.

When learning Bayesian network models the Bayesian way, MCMC is usually done over the model space of DAGs [11] or (simulated) essential graphs [5]. Usually the models are incrementally built by adding, removing or reversing arcs selected at random, thereby "moving" around in the state space. Unfortunately, by exploring the search space using moves defined solely in terms of single edges selected in a uniform fashion, we may easily get stuck in local maxima.

In this paper we suggest to traverse the model space in a more intelligent fashion by considering larger blocks. Rather than selecting edges uniformly, we select a set of vertices that are related to each other and propose a change of *how* these vertices are related by considering different edge assignments. We show that in doing so, MCMC is less prone to get stuck in local maxima compared to uniform edge proposals.

We proceed as follows: Section two introduces the basic theory for learning Bayesian networks based on the marginal likelihood criterion. In section three we discuss a Gibbs and a Metropolis-Hastings sampler for obtaining models from the posterior model distribution. Our new MCMC method is presented in section four, and in section five the pseudocode is given and some implementation issues are discussed. In section six we evaluate our method, and compare experimental results to those of a single edge MCMC approach. We draw conclusions in section seven.

2 Learning Models

A Bayesian network (BN) for the discrete random variables $\boldsymbol{X} = (X^1, \ldots, X^p)$ represents a joint probability distribution. It consists of a directed acyclic graph (DAG) M, called the model, where every vertex corresponds to a variable X^i, and a vector of conditional probabilities $\boldsymbol{\Theta}$, called the parameter, corresponding to that model. The joint distribution factors recursively according to M as $\Pr(\boldsymbol{X}|M, \boldsymbol{\Theta}) = \prod_{i=1}^{p} \Pr(X^i|\boldsymbol{\Pi}_i, \boldsymbol{\Theta}) = \prod_{i=1}^{p} \Theta_{X^i|\boldsymbol{\Pi}_i}$, where $\boldsymbol{\Pi}_i$ is the parent set of X^i in M. We consider $\boldsymbol{\Theta}$ a random (stochastic) vector with the following (prior) product Dirichlet distribution:

$$\Pr(\boldsymbol{\Theta}|M) = \prod_{i=1}^{p} \prod_{\boldsymbol{\pi}_i} \mathrm{Dir}(\Theta_{X^i|\boldsymbol{\pi}_i}|\boldsymbol{\alpha}) = \prod_{i=1}^{p} \prod_{\boldsymbol{\pi}_i} C(i, \boldsymbol{\pi}_i, \boldsymbol{\alpha}) \prod_{x^i} \Theta_{x^i|\boldsymbol{\pi}_i}^{\alpha(x^i, \boldsymbol{\pi}_i)-1},$$

with $C(i, \boldsymbol{\pi}_i, \boldsymbol{\alpha})$ the normalising factor:

$$C(i, \boldsymbol{\pi}_i, \boldsymbol{\alpha}) = \frac{\Gamma\Big(\alpha(\boldsymbol{\pi}_i)\Big)}{\prod_{x^i} \Gamma\Big(\alpha(x^i, \boldsymbol{\pi}_i)\Big)},$$

where $\boldsymbol{\alpha}$ is the vector of hyper parameters, $\alpha(\cdot)$ the function returning the prior *counts* for a particular configuration and $\Gamma(\cdot)$ the gamma function. The product Dirichlet captures the assumption of *parameter independence* [13], i.e. all parameters of the BN are distributed independently, each according to a Dirichlet distribution. We are given a multinomial data sample $\mathcal{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_c)$ with c i.i.d. cases, $\boldsymbol{d}_i = (x_i^1, \ldots, x_i^p)$,

for which the product Dirichlet is conjugate. This means that Bayesian updating is easy because the posterior once $\mathcal{D}$ has been taken into consideration is again product Dirichlet:

$$\Pr(\boldsymbol{\Theta}|\mathcal{D}, M) = \prod_{i=1}^{p} \prod_{\boldsymbol{\pi}_i} \mathrm{Dir}(\Theta_{X^i|\boldsymbol{\pi}_i}|\boldsymbol{\alpha} + \boldsymbol{s}) = \frac{\Pr(\mathcal{D}|\boldsymbol{\Theta}, M) \cdot \Pr(\boldsymbol{\Theta}|M)}{\Pr(\mathcal{D}|M)}$$
$$= \frac{\prod_{i=1}^{p} \prod_{x^i} \prod_{\boldsymbol{\pi}_i} \Theta_{x^i|\boldsymbol{\pi}_i}^{s(x^i,\boldsymbol{\pi}_i)} \cdot \Pr(\boldsymbol{\Theta}|M)}{\Pr(\mathcal{D}|M)},$$

where $\boldsymbol{s}$ is the vector of sufficient statistics from $\mathcal{D}$ and $s(\cdot)$ returns the counts for a particular configuration. We can now easily isolate the marginal likelihood $\Pr(\mathcal{D}|M)$:

$$\frac{\prod_{i=1}^{p} \prod_{x^i} \prod_{\boldsymbol{\pi}_i} \Theta_{x^i|\boldsymbol{\pi}_i}^{s(x^i,\boldsymbol{\pi}_i)} \cdot \Pr(\boldsymbol{\Theta}|M)}{\Pr(\boldsymbol{\Theta}|\mathcal{D}, M)} = \frac{\prod_{i=1}^{p} \prod_{\boldsymbol{\pi}_i} C(i, \boldsymbol{\pi}_i, \boldsymbol{\alpha}) \prod_{x^i} \Theta_{x^i|\boldsymbol{\pi}_i}^{\alpha(x^i,\boldsymbol{\pi}_i)+s(x^i,\boldsymbol{\pi}_i)-1}}{\prod_{i=1}^{p} \prod_{\boldsymbol{\pi}_i} C(i, \boldsymbol{\pi}_i, \boldsymbol{\alpha} + \boldsymbol{s}) \prod_{x^i} \Theta_{x^i|\boldsymbol{\pi}_i}^{\alpha(x^i,\boldsymbol{\pi}_i)+s(x^i,\boldsymbol{\pi}_i)-1}},$$

where everything except the normalising factors cancels out. By filling in the normalising factors we arrive at the marginal likelihood score, being a product of p terms, one term per variable, each term a function of the parent set of the variable in question:

$$\Pr(\mathcal{D}|M) = \prod_{i=1}^{p} \prod_{\boldsymbol{\pi}_i} \frac{\Gamma\Big(\alpha(\boldsymbol{\pi}_i)\Big)}{\Gamma\Big(\alpha(\boldsymbol{\pi}_i) + s(\boldsymbol{\pi}_i)\Big)} \prod_{x^i} \frac{\Gamma\Big(\alpha(x^i, \boldsymbol{\pi}_i) + s(x^i, \boldsymbol{\pi}_i)\Big)}{\Gamma\Big(\alpha(x^i, \boldsymbol{\pi}_i)\Big)}.$$

In [8] the marginal likelihood is derived from a Bayesian prediction (*prequential*) point of view using the chain rule $\Pr(\mathcal{D}|M) = \Pr(\boldsymbol{d}_1|M) \cdots \Pr(\boldsymbol{d}_c|\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{c-1}, M)$.

The posterior model distribution is computed by applying Bayes' law:

$$\Pr(M|\mathcal{D}) = \frac{\Pr(\mathcal{D}|M) \cdot \Pr(M)}{\sum_m \Pr(\mathcal{D}|m) \cdot \Pr(m)},$$

and then, if we are interested in some feature over models quantified by Δ, we can average:

$$\mathrm{E}[\Delta(M)|\mathcal{D}] = \sum_m \Delta(m) \cdot \Pr(m|\mathcal{D}).$$

The problem is however that we can't calculate the normalising factor $\Pr(\mathcal{D})$ due to the large number of models. A way of dealing with that problem is to apply MCMC over models; an alternative is described in [10].

3 MCMC for Model Learning

In this section we discuss two methods for learning models via MCMC; other MCMC approaches for learning BN models exist, see for instance [4]. An MCMC approach for model learning which does not employ the marginal likelihood criterion is given in [7].

Define for all $r = 1, \ldots, \frac{p\cdot(p-1)}{2}$ edges of M the random variables E_r with state space $\Omega_{E_r} = \{\leftarrow, \rightarrow, \not\leftrightarrow\}$, i.e. every edge of the graph can take on a direction, or can be absent. If the configuration of all edges forms a DAG, we can calculate the posterior joint distribution $\Pr(E_1, \ldots, E_{\frac{p\cdot(p-1)}{2}}|\mathcal{D})$.

3.1 Gibbs Sampling

The problem with the normalising factor mentioned in the previous section is solved by applying Gibbs sampling. Draw edges at iteration t from the full conditional given the data:

$$\begin{aligned}
e_1^t &\sim \Pr(E_1|e_2^{t-1}, \ldots, e^{t-1}_{\frac{p\cdot(p-1)}{2}}, \mathcal{D}) \\
&\vdots \\
e^t_{\frac{p\cdot(p-1)}{2}} &\sim \Pr(E_{\frac{p\cdot(p-1)}{2}}|e_1^t, \ldots, e^t_{\frac{p\cdot(p-1)}{2}-1}, \mathcal{D}) \\
e_1^{t+1} &\sim \Pr(E_1|e_2^t, \ldots, e^t_{\frac{p\cdot(p-1)}{2}}, \mathcal{D}) \\
&\vdots
\end{aligned}$$

where each draw is subject to the constraint that all edges together must form an acyclic graph. In order to draw edge E_l from the full conditional given the data we calculate:

$$\begin{aligned}
\Pr(E_l|\{e_{r\neq l}\}, \mathcal{D}) &= \frac{\Pr(e_1, \ldots, E_l, \ldots, e_{\frac{p\cdot(p-1)}{2}}|\mathcal{D})}{\sum_{e_l} \Pr(e_1, \ldots, e_l, \ldots, e_{\frac{p\cdot(p-1)}{2}}|\mathcal{D})} \\
&= \frac{\Pr(\mathcal{D}|e_1, \ldots, E_l, \ldots, e_{\frac{p\cdot(p-1)}{2}})}{\sum_{e_l} \Pr(\mathcal{D}|e_1, \ldots, e_l, \ldots, e_{\frac{p\cdot(p-1)}{2}})},
\end{aligned}$$

where a uniform model prior $\Pr(M)$ on the model space, and the denominator $\Pr(\mathcal{D})$ both cancel out. It is easy to simplify the ratio even more because of the marginal likelihood decomposition. When drawing an edge from the Gibbs sampler, say E_l, at most two terms are affected, namely the terms pertaining to the vertices of edge E_l. All remaining factors stay the same and cancel out in the ratio given above.

The Markov chain defined here is irreducible, because at every draw the state $\not\leftrightarrow$ is a possibility and this state never induces a cycle. Hence there is a non-zero probability of removing arcs that obstruct the addition of other edges in any direction in the graph (obstruct in the sense that the graph becomes cyclic). Also, since there is a non-zero probability of remaining in the current state the chain is aperiodic, and we conclude that the Markov chain is ergodic. Thus for $t \to \infty$ we have $(e_1, \ldots, e_{\frac{p\cdot(p-1)}{2}}) \sim$

$\Pr(E_1, \ldots, E_{\frac{p\cdot(p-1)}{2}} | \mathcal{D})$, i.e. the chain converges to the joint distribution over collections of edges that form models.

To approximate the expected value of model features, we use the empirical average (also called the Monte Carlo approximation):

$$\mathrm{E}[\Delta(M)|\mathcal{D}] \approx \frac{1}{N} \sum_{t=1}^{N} \Delta(m^t),$$

where N denotes the total number of samples from the Markov chain. Often this kind of averaging is done over features one can read directly off a model, e.g. Markov blanket features of vertices, but theoretically any statement that the model entails can be averaged.

3.2 Metropolis-Hastings Sampling

To our knowledge the model Gibbs sampler has not been proposed before in the form discussed in the previous section. However, an alternative set-up of MCMC sampling over models is to use MCMC Metropolis-Hastings sampling which is discussed in [11,9]. Here a *proposal distribution*, $q(\cdot)$, guides the incremental changes of the models by proposing which components to change. This proposal is produced by drawing $m^{t+1} \sim q(M|m^t)$. With probability:

$$\delta(m^{t+1}, m^t) = \min\left\{1, \frac{q(m^t|m^{t+1}) \Pr(m^{t+1}|\mathcal{D})}{q(m^{t+1}|m^t) \Pr(m^t|\mathcal{D})}\right\},$$

the proposal is accepted, otherwise $m^{t+1} = m^t$. For $t \to \infty$ models from the invariant distribution are obtained. Note that a uniform model prior, and $\Pr(\mathcal{D})$ cancel out in the acceptance ratio.

In all existing implementations we are aware of [5,11], the proposals pertain to components that correspond to single edges that are chosen in a uniform fashion. This is more or less similar to the Gibbs sampler presented above with the slight difference that the Gibbs sampler draws edges $E_1, \ldots, E_{\frac{p\cdot(p-1)}{2}}$ systematically. However, changing the visitation scheme does not invalidate the Gibbs sampler. The invariant distribution is reached no matter which visitation scheme is used, as long as all edges are sampled "infinitely often". With a random visitation scheme, both MCMC samplers behave very much the same way.

4 Markov Blanket Approach

The fundamental issue with a single edge approach is that from a MCMC perspective it is not the preferred way of decomposing the joint distribution. If we compare to a deterministic procedure where each single edge component is maximised on an individual basis, it would be quite obvious that this does not produce the best global solution per se. Although the MCMC samplers mentioned in the previous section, don't *maximise* individual edge components, the analogy holds because there is nevertheless a pressure

toward "good" or "maximum" solutions when drawing edges. A bad decomposition can make the sampler almost reducible, and we might easily get trapped in a local maximum. For the Gibbs sampler this means that the individual conditionals are very narrow (low variance), and although it is guaranteed that in the limit one will eventually escape from the local maximum, for all practical purposes this isn't useful.

In general for MCMC to perform well, the components should be as "self-contained" as possible in the sense that a component should be considered a unit that can't be split (see for instance Robert, Casella [12]). So, although the overall score of a model is a function of single edges, one should not necessarily reduce the sampling problem to one in terms of single edges. Instead we should try to discern (not necessarily disjoint) *blocks* or *sets* of edges that form a dependency relationship. These dependent edge variables of a DAG require special treatment because they inherently belong together.

The dependency between edges is expressed through the current parent sets of the vertices. By adding or removing a parent from a vertex through the addition, reversal or removal of an edge, the score of the vertex changes, and this change in score depends on the parent set: The probability of adding or removing an arc from, say X^1 to X^2, depends directly on all edges from the parents of X^2 to X^2 because the score of X^2 is a function of all its parents. Similarly, the probability of adding or removing an arc in the other direction depends directly on the edges from the parents of X^1 to X^1 because the score of X^1 is a function of its parents.

The dependency goes even further: the edge between X^1 and X^2 *indirectly* depends on the current edges to the vertices in the parent set and the child set of X^1 and X^2, etc. When the number of intermediate steps (arcs) is large between two edges connected through a series of arcs, the two edges are less dependent than when the number of intermediate steps is small; edges far apart are only *indirectly* relevant to each other. In this way different levels of dependency can be discerned depending on the distance between edges.

One way of defining component i is to group together currently dependent edges in the neighbourhood of vertex X^i. The strength of this edge dependency depends on how far from vertex X^i the edges are collected. We suggest to disregard edges that lie beyond the vertices of the Markov blanket of X^i. However, this set of edges only defines the *current* dependency relationship but not necessarily *the right* one which is exactly what is subject to learning. Therefore we suggest to also include in component i all edge variables *between* the vertices of the Markov blanket of X^i. Edges in the component are now perturbed by drawing (new) edge values, and because the edges were "close" to each other prior to the perturbation, the new value assignments may substantially change the edge dependencies within the component. This way, after several iterations, the component settles on a very likely assignment of all the edges.

Notice that the components overlap: many edge variables are part of several Markov blankets. Edge variables that are shared by many Markov blankets are crucial for determining the probability of the states of several other edge variables, i.e. for several edges they strongly influence the probability of adding, removing or reversing an arc.

Another form of edge dependency which is unrelated to the form of dependency we investigated here, is the dependency between variables (X^i) which stems from the domain the variables belong to. The *vertices* in the Markov blanket form a relatively

strong relationship, so proposing assignments $\leftarrow$ or $\rightarrow$ to edge variables in the Markov blanket is from a domain perspective a good heuristic; edges distant to each other are less likely to be associated.

The idea is as follows: Either shrink or grow the Markov blanket of a vertex by adding or removing vertices, *or* change the internal relationships between the vertices of the Markov blanket. The probability of doing the latter should be higher than the probability of doing the former; we want to try several Markov blanket configurations.

4.1 MCMC by Relevant Edges

We use a "Metropolis-within-Gibbs" MCMC, where Gibbs sampling and Metropolis-Hastings sampling are combined in order to sample models from the invariant model posterior. We define the set of *relevant edges*, $\boldsymbol{\mathcal{E}}_i$, of vertex X^i as $\boldsymbol{\mathcal{E}}_i = \boldsymbol{\mathcal{E}}'_i \cup \boldsymbol{\mathcal{E}}''_i$, where $\boldsymbol{\mathcal{E}}'_i$ is the set of all edge variables between vertices of the Markov blanket of X^i including X^i itself, i.e.:

$$\boldsymbol{\mathcal{E}}'_i = \{E = (X^s, X^t) | X^s, X^t \in \mathrm{MB}(X^i) \cup \{X^i\}\},$$

and $\boldsymbol{\mathcal{E}}''_i$ is the set of all edge variables between X^i and any other vertex of $\boldsymbol{X}$, i.e.:

$$\boldsymbol{\mathcal{E}}''_i = \{E = (X^i, X^t) | X^t \in \boldsymbol{X}\} \setminus \boldsymbol{\mathcal{E}}'_i.$$

We refer to $\boldsymbol{\mathcal{E}}'_i$ as the set of *currently relevant edges*, because it consists of all edges between all the relevant vertices given the Markov blanket as it is now. Observe that $\boldsymbol{\mathcal{E}}'_i$ actually captures the notion of a sub-graph. $\boldsymbol{\mathcal{E}}''_i$ is referred to as the set of *potentially relevant edges* because it consists of edges between a vertex that is currently relevant and vertices that may become relevant. In figure 1 the two relevance sets are illustrated.

Using block Gibbs sampling we now draw an instantiation of the edges in the set of relevant edges per vertex:

$$\varepsilon_1^t \sim \Pr(\boldsymbol{\mathcal{E}}_1 | \bar{\varepsilon}_1, \mathcal{D}) \quad \cdots \quad \varepsilon_p^t \sim \Pr(\boldsymbol{\mathcal{E}}_p | \bar{\varepsilon}_p, \mathcal{D}) \quad \varepsilon_1^{t+1} \sim \Pr(\boldsymbol{\mathcal{E}}_1 | \bar{\varepsilon}_1, \mathcal{D}) \quad \cdots$$

where $\bar{\varepsilon}_j$ is the *current* configuration of the edges in the complement of the set $\boldsymbol{\mathcal{E}}_j$.

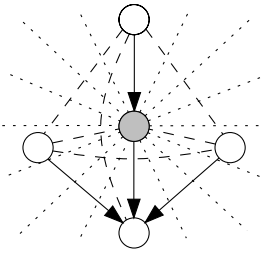

Fig. 1. The set of relevant edges of the shaded vertex. The solid lines (arcs) connect the *vertices* belonging to the Markov blanket. Dashed *and* solid lines indicate the set of currently relevant edges, and dotted lines indicate the set of potentially relevant edges.

The Gibbs sampler visits every vertex, thus every edge will eventually be part of a *potentially relevant edge set*; we can't guarantee that it will be part a *currently relevant edge set* however, but as long as all edges *are* considered, we have that $(e_1, \ldots, e_{\frac{p\cdot(p-1)}{2}}) \sim \Pr(E_1, \ldots, E_{\frac{p\cdot(p-1)}{2}}|\mathcal{D})$ for $t \to \infty$.

Each draw from the above Gibbs sampler is performed by a Metropolis-Hastings sampler. The proposal distribution is a mixture distribution, where one component, $f(\cdot)$, deals with the edges in the currently relevant edge set, and the other component, $g(\cdot)$, deals with the edges in the potentially relevant edge set, that is:

$$q(\boldsymbol{\mathcal{E}}^{t+1}|\boldsymbol{\mathcal{E}}^t) = w \cdot f(\boldsymbol{\mathcal{E}}^{t+1}|\boldsymbol{\mathcal{E}}'^t) + (1-w) \cdot g(\boldsymbol{\mathcal{E}}^{t+1}|\boldsymbol{\mathcal{E}}''^t),$$

where $0 < w < 1$ determines the mixture weights. When $f(\cdot)$ is applied, values for the edges in the currently relevant set are drawn, i.e. MCMC is run in order to obtain the posterior distribution of those edges. The cardinality of the set of relevant edges is kept fixed once the Metropolis-Hasting sampler is applied—we merely assign (new) values to the edge variables. Hence, the set of relevant edges is determined before entering the Metropolis-Hastings sampler, and does not change until control is given back to the overall Gibbs sampler. For $g(\cdot)$ the same holds, but here assignments are considered to the variables in the potentially relevant set. The distribution $f(\cdot)$ produces uniform proposals by selecting an edge $E_r \in \boldsymbol{\mathcal{E}}'$ with probability $1/|\boldsymbol{\mathcal{E}}'|$. Depending on the current value of e_r, a state change is proposed to one of the (at most) two alternatives with probability 0.5. E.g. if $E_r^t = \not\leftrightarrow$ then either $E_r^{t+1} = \rightarrow$ or $E_r^{t+1} = \leftarrow$ is proposed. For the distribution $g(\cdot)$ the same holds, but here we have $E_r \in \boldsymbol{\mathcal{E}}''$ with probability $1/|\boldsymbol{\mathcal{E}}''|$. Notice that edges not in either of these two sets remain unchanged.

For Metropolis-Hastings, the acceptance probability depends on the proposal fraction $q(\boldsymbol{\mathcal{E}}^t|\boldsymbol{\mathcal{E}}^{t+1})/q(\boldsymbol{\mathcal{E}}^{t+1}|\boldsymbol{\mathcal{E}}^t)$, which is required to ensure detailed balance and hence invariance. For both the mixture in the numerator and in the denominator, the weights are the same, the conditional distributions select edges with equal probability and there is always the same number of alternative edge assignments, i.e. the distributions are uniform, hence, the ratio cancels out.

The proposal distribution of the sampler will with a non-zero probability propose a state change to any edge in the relevant edge set, which guarantees irreducibility. With a non-zero probability it will remain in the current state for any edge implying aperiodicity. We may thus conclude that the Metropolis-Hastings sampler will in the limit return realisations from the invariant distribution $\Pr(\boldsymbol{\mathcal{E}}_l|\bar{\boldsymbol{\mathcal{E}}}_l, \mathcal{D})$, i.e. realisations for the edges in $\boldsymbol{\mathcal{E}}_l$ given all other edges.

By introducing the *currently relevant edge* set, which is dealt with by $f(\cdot)$, we can through the weight w vary how much "attention to pay" to the configuration of the edge variables in the Markov blanket of the current vertex. Note that because edges in the currently relevant edge set are proposed uniformly, edge variables that are part of several Markov blankets are sampled relatively often. This entails that edges in dense regions of the graph are sampled more often than edges in less dense regions. Edge variables in dense regions are more dependent on each other, hence we sample more in that region because it is easy to get "stuck" there since the edges constrain each other strongly. Sampling more often in dense regions corresponds to putting more effort into avoiding getting trapped in sub-optimal assignments to the edge variables.

4.2 Covered Arcs

In recent years so-called *inclusion-driven* learning approaches [3] for model selection have emerged, in which the essential graph space is traversed by respecting the *inclusion order*. Unfortunately, it is inefficient to score essential graphs directly while traversing the search space. As an alternative, one can *simulate* the essential graph space by repeatedly reversing arcs that have the same parent set—the covered edges. Although our MCMC method is not really an inclusion driven approach, we may still profit from the essential graph simulation idea. By doing covered arc reversal often enough all DAGs representing the same set of independences are reached. All these DAGs have the same marginal likelihood score, yet the individual score of the vertices (recall that the marginal likelihood score decomposes) is different for equivalent DAGs. By reversing covered arcs we may increase the probability of assigning alternative values to dependent edge variables that prior to reversal perhaps obstruct each other. We employ the Repeated Covered Arc Reversal (RCAR) algorithm [3] to perform the covered arc reversals. RCAR takes an argument, that determines the maximum number of times covered arcs will be reversed. A value between 4 and 10 should suffice, and in particular 10 seems to work well.

5 Implementation Issues

In figure 2 the pseudocode of the Markov blanket MCMC (MB-MCMC) algorithm is given. Line 2 determines the component to pay attention to, here a systematic sweep is shown, but a random choice is also possible. Line 3 calls the algorithm for reversing covered arcs. Lines 4–5 determines the edges to consider, and in lines 7–9 the edges are drawn from the sets of relevant edges. The proposals are accepted or rejected in line 11–12. In line 13 the configuration of all edges is recorded, i.e. here the actual models from the posterior are saved. One may decide to sub-sample the Markov chain of models by only recording the draws once in a while.

Algorithm MB-MCMC(k, w)

1 **for** $r \leftarrow 0$ **to** ∞
2 $\quad i \leftarrow (r \bmod p) + 1$
3 $\quad \text{RCAR}(10)$
4 $\quad \boldsymbol{\mathcal{E}}'_i \leftarrow \{E = (X^s, X^l) | X^s, X^l \in \text{MB}(X^i) \cup \{X^i\}\}$
5 $\quad \boldsymbol{\mathcal{E}}''_i \leftarrow \{E = (X^i, X^l) | X^l \in \boldsymbol{X}\} \setminus \boldsymbol{\mathcal{E}}'_i$
6 $\quad$ **for** $t \leftarrow 0$ **to** k
7 $\quad\quad$ **draw** $u \sim \mathcal{U}[0, 1]$
8 $\quad\quad$ **if** $u < w$ **and** $\boldsymbol{\mathcal{E}}'_i \neq \varnothing$ **then draw** $\boldsymbol{\varepsilon}_i^{t+1} \sim f(\boldsymbol{\mathcal{E}}_i | \boldsymbol{\varepsilon}_i'^t)$
9 $\quad\quad\quad$ **else draw** $\boldsymbol{\varepsilon}_i^{t+1} \sim g(\boldsymbol{\mathcal{E}}_i | \boldsymbol{\varepsilon}_i''^t)$
10 $\quad\quad$ $\delta \leftarrow \Pr(\mathcal{D} | \boldsymbol{\varepsilon}_i^{t+1}) / \Pr(\mathcal{D} | \boldsymbol{\varepsilon}_i^t)$
11 $\quad\quad$ **draw** $u \sim \mathcal{U}[0, 1]$
12 $\quad\quad$ **if** $u \geq \min\{1, \delta\}$ **then** $\boldsymbol{\varepsilon}_i^{t+1} \leftarrow \boldsymbol{\varepsilon}_i^t$
13 $\quad$ $\text{RECORD}(e_1, \ldots, e_{\frac{p \cdot (p-1)}{2}})$

Fig. 2. Pseudocode of the Markov blanket MCMC

The algorithm takes two arguments: k determines the number of times the Metropolis-Hastings sampler is run, and w determines the probability of changing the internal configuration of a component vs. adding or removing new vertices. Parameter k need not be large for the overall invariant model distribution to be reached, i.e. the Metropolis-Hastings sampler need not converge at every call. In fact we have found it to be beneficial for the convergence rate to assign k a small value; too large a value may lead to premature convergence. In our experiments we have set $k = 5$, and $w = 0.95$.

When every vertex is assigned a cache that keeps the sufficient statistics indexed by the parent set, we may drastically improve the speed of MCMC by querying the cache before querying the data. We have implemented the Markov blanket sampler in C++ using STL, and for the experiments in the next section we were able to reach what we believe are the invariant distributions in less than 10 minutes on a 2 GHz machine.

6 Evaluation

We considered two BNs for the experiments: the ALARM BN with 37 vertices and 46 arcs [1], and the Insurance BN with 27 vertices and 52 arcs [2]. We used the BDeu metric for the counts $\boldsymbol{\alpha}$ with an equivalent sample size of 1. All experiments were run for 1,000,000 iterations. As convergence diagnostic we monitored the number of edges as suggested in for instance [6]. We compared the Markov blanket MCMC with $e\text{MC}^3$, a single edge MCMC sampler that also employs the RCAR algorithm.

In figure 3a the results of the ALARM network are illustrated. With 1000 samples, we see that two independent runs of the MB-MCMC both converge towards models with about 50–53 edges. There is no significant difference in the convergence behaviour. For $e\text{MC}^3$ two runs produce different behaviour and result in models with 68–77 edges. For 5000 records similar observations hold, but overall the number of edges is lower: 45–51 for MB-MCMC and 57–70 for $e\text{MC}^3$. We notice that $e\text{MC}^3$ seems sensitive to the starting point of the chain. To show this more clearly, we ran both samplers starting from the empty graph, and from the actual ALARM graph for 7000 samples. For the 7000 records we would expect that the number of edges on average should converge to 46, i.e. there is enough data to support the data generating model. For MB-MCMC, both chains converge towards models with 44–50 edges. The most frequently sampled model is similar to the ALARM network ± 2 arcs. For $e\text{MC}^3$ there is a big difference. The chain started from the actual network stays at around 50–55 edges, but the chain started from the empty graph gets stuck at 63–70. The most frequently sampled model is in both situations less similar to the actual ALARM network than in the MB-MCMC case (excess of ± 10 and ± 25 arcs).

Next we consider results of the Insurance network in figure 3b. We would like to note that the association between several parent-child variables in the Insurance network is rather weak and that even for large data sets these associations will be deemed absent by the marginal likelihood score. For 500 records the MB-MCMC converges to an invariant distribution where models are sampled with 36–40 edges. The two runs meet at around 150,000 iterations. For $e\text{MC}^3$ however, the two chains don't quite agree in the number of edges: somewhere between 37–46. We also ran both samplers beginning from the empty and the actual Insurance graph. For MB-MCMC both starting points produce models with 45–47 edges. Also here we see that $e\text{MC}^3$ is sensitive to

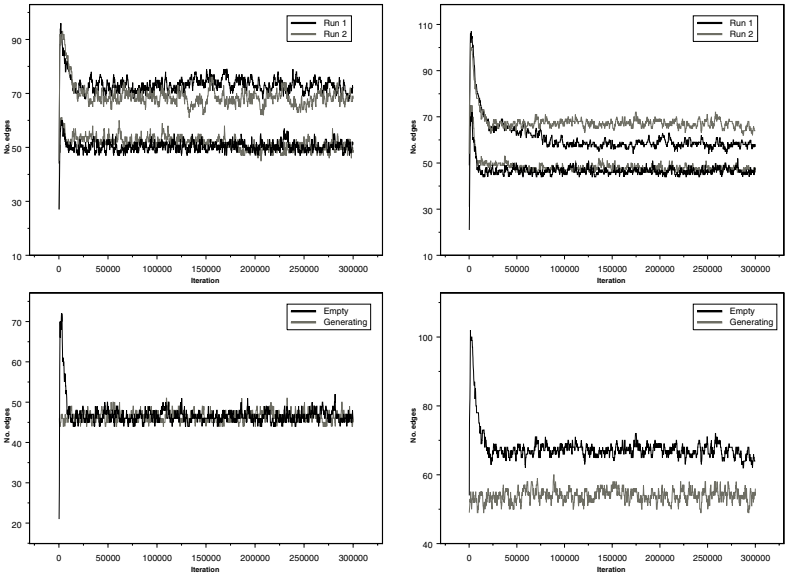

Fig. 3a. ALARM network. *Top:* Convergence behaviour given 1000 (*left*) and 5000 (*right*) records for two independent runs. The lower lines are from the Markov blanket MCMC, and the upper lines from $e\text{MC}^3$. *Bottom:* Convergence behaviour of the Markov blanket MCMC (*left*) and $e\text{MC}^3$ (*right*) given 7000 records starting from the empty and the data generating model.

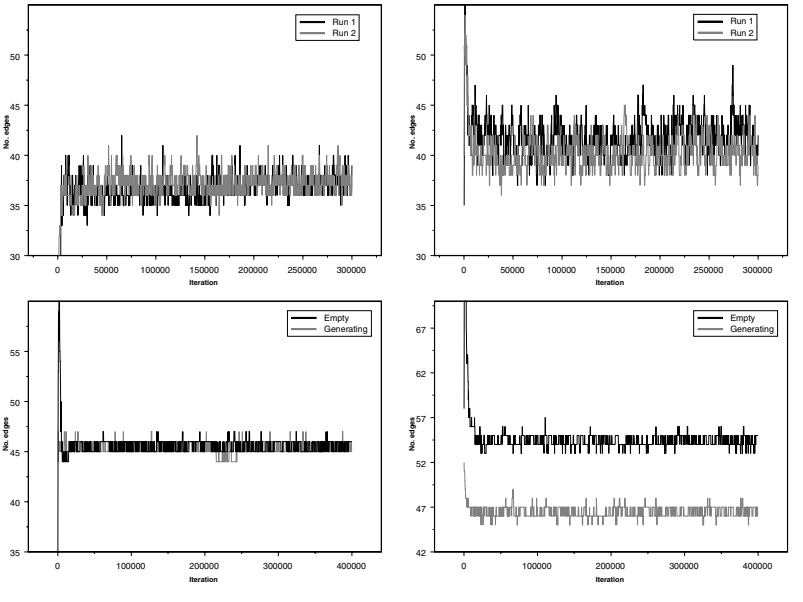

Fig. 3b. Insurance network. *Top:* Convergence behaviour given 500 records for two independent runs (common y-axis) of the Markov blanket MCMC (*left*) and $e\text{MC}^3$ (*right*). *Bottom:* Convergence behaviour of the Markov blanket MCMC (*left*) and $e\text{MC}^3$ (*right*) given 10,000 records starting from the empty and the data generating model.

the initial model. Starting from the data generating model, the sampler converges to an invariant distribution where models with 45–47 edges are sampled. Starting from the empty graph, models with 54–56 edges are sampled. We see that even with 10,000 records, there is not enough information in the data sample to support the 52 arcs in the data generating Insurance network. Also notice that the variability of the plots for 500 records is larger than for 10,000 records. This is to be expected because there is no pronounced "best" model with merely 500 records.

7 Conclusion

We have proposed a new MCMC method for learning Bayesian network models. By defining components of MCMC in terms of all edges in the Markov blankets of the vertices, we group relatively strongly dependent edges together. This effectively means that edges in dense regions of a model are sampled more often than in less dense regions.

Our experiments on the ALARM and the Insurance networks show that this Markov blanket decomposition performs better than the naive MCMC approach, where all edges are sampled equally often. The chain mixes faster, and it is less sensitive to the departure model. This indicates that MB-MCMC is less prone to getting stuck in local maxima.

References

1. I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proc. of the European Conf. on AI in Medicine*, 1989.
2. J. Binder, D. Koller, S. J. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.
3. R. Castelo and T. Kocka. On inclusion-driven learning of Bayesian networks. *J. of Machine Learning Research*, 4:527–574, 2003.
4. N. Friedman and D. Koller. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1–2):95–125, 2003.
5. P. Giudici and R. Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50(1):127–158, 2003.
6. P. Giudici and P. Green. Decomposable graphical gaussian model determination. *Biometrika*, 86(4):785–801, 1999.
7. P. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1998.
8. D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.
9. T. Kocka and R. Castelo. Improved learning of Bayesian networks. In D. Koller and J. Breese, editors, *Proc. of the Conf. on Uncertainty in AI*, pages 269–276, 2001.
10. D. Madigan and A. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *J. of the Am. Stat. Assoc.*, 89:1535–1546, 1994.
11. D. Madigan and J. York. Bayesian graphical models for discrete data. *Intl. Statistical Review*, 63:215–232, 1995.
12. C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer-Verlag, 3rd edition, 2002.
13. D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.