Stat Comput (2017) 27:115–130
DOI 10.1007/s11222-015-9611-4

Learning discrete decomposable graphical models via constraint optimization

Tomi Janhunen[1] · **Martin Gebser**[1,2] · **Jussi Rintanen**[1,3] · **Henrik Nyman**[4] · **Johan Pensar**[4] · **Jukka Corander**[5]

Received: 9 March 2015 / Accepted: 1 November 2015 / Published online: 11 November 2015

Abstract Statistical model learning problems are traditionally solved using either heuristic greedy optimization or stochastic simulation, such as Markov chain Monte Carlo or simulated annealing. Recently, there has been an increasing interest in the use of combinatorial search methods, including those based on computational logic. Some of these methods are particularly attractive since they can also be successful in proving the global optimality of solutions, in contrast to stochastic algorithms that only guarantee optimality at the limit. Here we improve and generalize a recently introduced constraint-based method for learning undirected graphical models. The new method combines perfect elimination orderings with various strategies for solution pruning and offers a dramatic improvement both in terms of time and memory complexity. We also show that the method is capable of efficiently handling a more general class of models, called stratified/labeled graphical models, which have an astronomically larger model space.

Keywords Graphical models · Structure learning · Constraint programming · Satisfiability · MAXSAT · Answer set programming

✉ Jukka Corander
Jukka.Corander@helsinki.fi

Tomi Janhunen
Tomi.Janhunen@aalto.fi

Martin Gebser
Martin.Gebser@aalto.fi

Jussi Rintanen
Jussi.Rintanen@aalto.fi

Henrik Nyman
Henrik.Nyman@abo.fi

Johan Pensar
jopensar@abo.fi

[1] Department of Computer Science, Aalto University, Espoo, Finland

[2] University of Potsdam, Potsdam, Germany

[3] Griffith University, Brisbane, Australia

[4] Department of Mathematics and Statistics, Åbo Akademi University, Åbo, Finland

[5] Department of Mathematics and Statistics, University of Helsinki, Helsinki, Finland

1 Introduction

Score-based optimization or structural learning of statistical models is typically performed over finite classes of models, where the topology of the search space poses a challenge for building an algorithm that can efficiently traverse across hills and valleys shaping a multimodal target function to be optimized. In particular in Bayesian model learning it is frequently possible to score individual models based on data using analytical formulas for the log unnormalized posterior, either exactly or approximately. Algorithms for stochastic simulation, such as Markov chain Monte Carlo (MCMC) and simulated annealing, represent popular ways to identify posterior optimal models. Learning of undirected graphical models, also known as Markov networks, is a widely considered application of such tools (Corander et al. 2008; Dellaportas and Forster 1999; Giudici and Castello 2003; Giudici and Green 1999; Madigan and Raftery 1994). One attractive property of these algorithms is their consistency at the limit, i.e., they are known to identify an optimal model with certainty as the number of iterations goes towards infinity. On the other hand, this does not guarantee anything concerning their performance for a finite number of iterations.

Statistical model learning has traditionally not been considered from the perspective of Boolean propositional logic, given the apparent separation of the two research traditions. However, learning of Bayesian networks and Markov networks has recently attained considerable interest in the computational logic and machine learning communities, and subsequently the use of various constraint-based methods, such as integer programming, maximum satisfiability, and answer set programming, has been proposed (Bartlett and Cussens 2013; Berg et al. 2014; Corander et al. 2013; Cussens 2008; Parviainen et al. 2014).

Graphical models are an essential tool when considering modular representations of multivariate systems. Despite of the versatility of Markov networks to encode the dependence structure over a set of discrete variables, certain more subtle types of independencies cannot be expressed in such models. The dependence structure induced by Markov networks may be unnecessarily stringent in that it can only convey conditional independencies that hold in the entire outcome space of the variables involved. This has motivated the development of new classes of models. Using the theory of log-linear models for contingency tables, Markov networks have been generalized in a number of ways (Corander 2003; Eriksen 1999, 2005; Højsgaard 2003, 2004). The common basis of these models is that conditional independence is augmented by an independence that holds only in a subset of the joint state space of the variables included in a particular condition. A similar construction has been proposed for Bayesian networks in context-dependent Bayesian networks (Boutilier et al. 1996; Friedman and Goldszmidt 1996; Koller and Friedman 2009; Pensar et al. 2015).

Recently, a class of Markov networks that belong to the general family of context-specific graphical models was introduced (Nyman et al. 2014). Termed as stratified graphical models (alternatively called as labeled Markov networks), this is the first class of undirected graphical models that allows for the graphical representation of both conditional and context-specific independencies, while simultaneously enabling fully Bayesian learning under an explicit factorization of the joint probability distribution. A non-reversible Markov chain Monte Carlo (MCMC) algorithm has been utilized for learning the maximum a posteriori (MAP) model from data by Nyman et al. (2014).

In this paper, we develop further a constraint-based learning method introduced for ordinary Markov networks (Corander et al. 2013) by significantly extending applicability with respect to network size and by also allowing context-specific independence restrictions. The earlier method is not directly applicable to larger Markov networks or labeled networks due to the impractically large number of possible model elements that need to be explicitly considered. This prompted us to develop several formal pruning criteria. They allow for reducing the number of relevant elements dramatically, so that constraint-based methods become applicable. Our pruning criteria are based on statistical arguments, such that they act consistently in model rejection as the number of samples available for model learning tends towards infinity. In addition we replace the balancing condition and maximum spanning tree considerations in (Corander et al. 2013) by a perfect elimination ordering, which is much better suited for obtaining a compact representation in terms of model constraints. We demonstrate the feasibility of this approach with examples that have model cardinalities ranging from very large to an astronomic size.

The structure of the paper is as follows. We start by introducing Markov networks and labeled Markov networks in Sect. 2. Section 3 presents the principles we use for pruning the set of candidate cliques to improve efficiency. A characterization of chordal graphs in terms of perfect elimination orderings is given in Sect. 4. Section 5 provides a constraint-based representation of ordinary Markov networks and a respective extension to labeled networks. An experimental evaluation of the proposed method is given in Sect. 6, and Sect. 7 concludes the paper.

2 Markov networks and their generalization

In the following we provide a brief introduction to graphical models and summarize results derived for context-dependent (labeled) Markov networks (Nyman et al. 2014). For a more comprehensive presentation of the theory of probabilistic graphical models see the standard references (Koller and Friedman 2009; Lauritzen 1996; Whittaker 1990). An undirected graph $G = \langle N, E \rangle$ consists of a set N of *nodes* standing for random variables and a set $E \subseteq N \times N$ of *undirected edges*. Two nodes δ and γ are *adjacent* in G if $\{\delta, \gamma\} \in E$. A *path* is a sequence of nodes such that every two consecutive nodes in the sequence are adjacent. Two sets A and B of nodes are *separated* by a third set D of nodes if every path between a node in A and a node in B contains at least one node in D. An undirected graph is *chordal* if, for every path $X_1, \ldots, X_n, X_1$ with $n \geq 4$, there are two non-consecutive nodes on the path connected by an edge. A *clique* c is a set of nodes such that every pair of nodes in c is adjacent. Given the set C of maximal cliques in a chordal graph, the multiset of *separators* can be obtained as intersections of the cliques ordered by a spanning tree of the respective clique graph (Golumbic 1980). Some details of this construction are given in Sect. 5.

A Markov network consists of a graph $G = \langle N, E \rangle$ and a joint distribution P_N over the variables X_N. The graph specifies the dependence structure of the variables and P_N factorizes according to G (see below for chordal graphs). Given G it is possible to ascertain whether two sets X_A and X_B of variables are conditionally independent given another

set X_D of variables, due to the global Markov property

$$X_A \perp X_B \mid X_D, \text{ if } D \text{ separates } A \text{ from } B \text{ in } G.$$

For a chordal graph G the probability of a joint outcome $x_N \in \mathscr{X}_N$, where $\mathscr{X}_N$ denotes the state space of the variables X_N, factorizes as

$$P_N(x_N) = \frac{\prod_{c\in C} P_c(x_c)}{\prod_{s\in S} P_s(x_s)}.$$

We assume that all outcomes have strictly positive probabilities and that the prior distribution for the model parameters factorizes with respect to the graph. Then, the marginal likelihood of a dataset $\mathbf{X}$ given a chordal graph (Dawid and Lauritzen 1993) can be written as

$$P(\mathbf{X} \mid G) = \frac{\prod_{c\in C} P_c(\mathbf{X}_c)}{\prod_{s\in S} P_s(\mathbf{X}_s)}. \tag{1}$$

By employing a Dirichlet distribution to assign prior probabilities to outcomes $x_c \in \mathscr{X}_c$, the terms $P_c(\mathbf{X}_c)$ and $P_s(\mathbf{X}_s)$ can be calculated analytically. These properties are inherited by the class of *decomposable labeled Markov networks*, which allow a full factorization of the probability for all joint outcomes including context-specific independencies within maximal cliques. In addition context-specific independencies can be expressed in such Markov networks through *labels*. The assumption of decomposability of labeled networks permits an analytical scoring function, which has been customarily used also for ordinary Markov networks in the Bayesian approaches to model learning.

Definition 1 *(Label)* Let (G, P_N) be a Markov network. For all $\{\delta, \gamma\} \in E$, let $L_{\{\delta,\gamma\}}$ denote the set of nodes adjacent to both δ and γ. For a non-empty $L_{\{\delta,\gamma\}}$, define the label of the edge $\{\delta, \gamma\}$ as the subset $\mathscr{L}_{\{\delta,\gamma\}}$ of outcomes $x_{L_{\{\delta,\gamma\}}} \in \mathscr{X}_{L_{\{\delta,\gamma\}}}$ for which X_δ and X_γ are independent given $X_{L_{\{\delta,\gamma\}}} = x_{L_{\{\delta,\gamma\}}}$, i.e.,

$$\mathscr{L}_{\{\delta,\gamma\}} = \left\{x_{L_{\{\delta,\gamma\}}} \in \mathscr{X}_{L_{\{\delta,\gamma\}}} : X_\delta \perp X_\gamma \mid X_{L_{\{\delta,\gamma\}}} = x_{L_{\{\delta,\gamma\}}}\right\}.$$

A label establishes a context in which a specific conditional independence holds. An edge $\{\delta, \gamma\}$ is said to be labeled if $\mathscr{L}_{\{\delta,\gamma\}}$ is well-defined and non-empty. Restricting a Markov network with the collection L of all labels induces a labeled Markov network.

Definition 2 *(Labeled Markov network)* A labeled Markov network is defined by the triple (G, L, P_N), where G is the underlying graph, L equals the joint collection of all labels $\mathscr{L}_{\{\delta,\gamma\}}$ for the edges of G, and P_N is a joint distribution over X_N that factorizes according to the restrictions imposed by G and L.

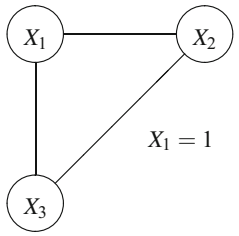


Fig. 1 Labeled graph encompassing the context-specific independence $X_2 \perp X_3 | X_1 = 1$

The pair (G, L) constitutes a labeled graph G_L. Figure 1 shows the graphical representation of a labeled Markov network containing three conditionally dependent variables with the context-specific independence $X_2 \perp X_3 | X_1 = 1$.

Imposing certain restrictions to the labeled graph will enable the factorization of $P(\mathbf{X} \mid G_L)$ according to (1) as well as an analytic expression of $P_c(\mathbf{X}_c)$ and $P_s(\mathbf{X}_s)$.

Definition 3 *(Decomposable labeled graph)* Let (G, L) be a labeled graph such that G is chordal. Further, let E_L denote the set of labeled edges, E_c the set of all edges in a maximal clique c, and E_S the set of all edges in the separators of G. The labeled graph is *decomposable* if

$$E_L \cap E_S = \emptyset$$

and, for all $c \in C$,

$$E_L \cap E_c = \emptyset \quad \text{or} \quad \bigcap_{\{\delta,\gamma\}\in E_L\cap E_c} \{\delta, \gamma\} \neq \emptyset.$$

A labeled graph is decomposable if G is chordal, there are no labels for edges in separators, and in each maximal clique all labeled edges have at least one node in common. A labeled Markov network with a decomposable labeled graph is a *decomposable labeled Markov network*. The first two restrictions will allow for P_N to be factorized according to (1). The third restriction will enable an ordering $(X_1, \ldots, X_{|c|})$ of the variables in X_c, such that the variables in $\{X_1, \ldots, X_{|c|-1}\}$ can be considered dependent on each other regardless of the context, and the variable $X_{|c|}$ may or may not be independent of a subset of $\{X_1, \ldots, X_{|c|-1}\}$ depending on the context.

If $X_{|c|}$ is the variable corresponding to the node in common to all labeled edges in a maximal clique, the variables $\{X_1, \ldots, X_{|c|-1}\}$ can be considered parents of $X_{|c|}$, denoted by $\Pi_{|c|}$. In a Markov network each outcome of the variables in $\Pi_{|c|}$ would induce a specific conditional distribution for $X_{|c|}$. However, for labeled Markov networks some outcomes of $\Pi_{|c|}$ may be grouped together, with all outcomes in a group inducing the same conditional distribution for $X_{|c|}$ (Boutilier et al. 1996). This creates a partition of the outcome space of $\Pi_{|c|}$, where each cell in the partition forms a *distinguishable parent combination* for $X_{|c|}$.

For decomposable labeled graphs, $P_c(\mathbf{X}_c)$ can be calculated (Nyman et al. 2014) using the formula

$$\prod_{j=1}^{|c|}\prod_{l=1}^{q_j}\frac{\Gamma\left(\sum_{i=1}^{k_j}\alpha_{jil}\right)}{\Gamma\left(n(\pi_j^l)+\sum_{i=1}^{k_j}\alpha_{jil}\right)}\prod_{i=1}^{k_j}\frac{\Gamma\left(n(x_j^i \mid \pi_j^l)+\alpha_{jil}\right)}{\Gamma(\alpha_{jil})}, \tag{2}$$

where q_j is the number of *distinguishable parent combinations* for variable X_j (i.e., there are q_j distinct conditional distributions for variable X_j), k_j is the number of possible outcomes for variable X_j, α_{jil} is the hyperparameter in the Dirichlet distribution corresponding to the outcome i of variable X_j given that the parental combination of X_j equals l, $n(\pi_j^l)$ is the number of observations of the combination l for the parents of variable X_j, and finally, $n(x_j^i \mid \pi_j^l)$ is the number of observations, where the outcome of variable X_j is i given that the observed outcome of the parents of X_j equals l. Note that for $X_{|c|}$ a parent configuration l is not necessarily comprised of a single outcome of $\Pi_{|c|}$, but rather the set of outcomes inducing the same conditional distribution for $X_{|c|}$.

The following choice of hyperparameters for the Dirichlet distribution is motivated by the desideratum that the ordering of the variables in $\Pi_{|c|}$ should be irrelevant for statistical learning of the model structure:

$$\alpha_{jil} = \alpha_{jl} = \frac{|\mathscr{X}_c| \cdot \lambda_{jl}}{\pi_j \cdot k_j},$$

where π_j is the total number of possible outcomes for the parents of variable X_j and k_j is the number of possible outcomes for variable X_j. Further, λ_{jl} equals the number of outcomes for the parents of variable X_j in *group* l with an equivalent effect on X_j, if X_j is the last variable in the ordering. Otherwise, λ_{jl} equals one. The distribution $P_s(\mathbf{X}_s)$ can also be calculated using (2). Note that our choice of hyperparameters need not lead to a hyper-consistent prior distribution, as defined in Dawid and Lauritzen (1993). However, research in the machine learning field has clearly demonstrated that Dirichlet priors which are not necessarily hyper-consistent do in fact allow for the correct underlying dependence structure to be better learned than priors which ensure consistency. Marginal likelihood under a Dirichlet prior with constant hyperparameters has been shown to be very robust against scoring models with consistent priors, such as the BDe score (Silander et al. 2007, 2008, 2010).

Given a labeled Markov network, the marginal likelihood of a dataset can be calculated by combining (1) and (2), and for practical purposes we consider only the logarithmic value $\log P(\mathbf{X}|G_L)$. Introducing the notation $v(c, L) = \log P_c(\mathbf{X}_c)$, which for non-empty label sets is dependent on the value of L, the log marginal likelihood can be written as

$$\log P(\mathbf{X}|G_L) = \sum_{c\in C} v(c, L) - \sum_{s\in S} v(s, L). \tag{3}$$

The learning problem we consider consists of finding the labeled graph G_L that maximizes the posterior distribution

$$P(G_L|\mathbf{X}) = \frac{P(\mathbf{X}|G_L)P(G_L)}{\sum_{G_L\in\mathscr{G}} P(\mathbf{X}|G_L)P(G_L)},$$

which can be reduced to identifying the model that optimizes $P(\mathbf{X}|G_L)P(G_L)$. Here $\mathscr{G}$ denotes the set of all graphs under consideration and $P(G_L)$ is the prior probability assigned to G_L. Here we use a prior penalizing dense graphs

$$P(G_L) \propto 2^{|N|-f},$$

where $|N|$ is the number of nodes in G and f is the number of free parameters in a distribution P_N obeying G. This choice of prior is motivated by the fact that adding a label to a sparse graph often induces a context-specific independence in a larger context than adding a label to a dense graph. The value $2^{f-|N|}$ is a numerically convenient approximation of the number of unique dependence structures that can be derived by adding labels to G.

Different types of MCMC methods are often applied to model optimization problems. While offering statistical consistency for the resulting estimates, a drawback of such an approach is that when the model space grows the number of iterations required to sample an optimal model even once may become prohibitively high. In addition, there are no general guarantees that a finite sample estimate corresponds to a true posterior optimal model. These aspects are particularly relevant when considering labeled Markov networks as the size of the model space is astronomical even for a moderate number of nodes. For Markov networks the model space grows according to the formula $2^{d(d-1)/2}$, where d is the number of nodes in the graph. For decomposable labeled Markov networks over a set of binary nodes a lower bound for the cardinality of the model space is given by

$$d \cdot 2^{2^{d-2}\cdot(d-1)} - d + 1 - \frac{d(d-1)}{2}\cdot(2^{2^{d-2}} - 1),$$

which is the number of different label combinations for a maximal clique with d nodes while satisfying the restrictions of a decomposable labeled graph. Using this result, Table 1 shows the size of the model space relative to different numbers of nodes in Markov networks and labeled Markov networks.

Table 1 Size of model space given the number of nodes in the system

Nodes	Markov networks	Labeled Markov networks
3	8	>37
4	64	>16291
5	1024	$>2.15 \times 10^{10}$
6	32768	$>7.25 \times 10^{24}$
7	2.10×10^{6}	$>4.39 \times 10^{58}$
8	2.68×10^{8}	$>5.81 \times 10^{135}$

3 Filtering clique candidates

To apply a constraint-based learning method of the type introduced in Corander et al. (2013) it is necessary to construct a database containing the scores (log unnormalized posteriors) $v(c, L)$ for all possible clique candidates c (for Markov networks) and the respective labelings L of edges (for labeled Markov networks). Due to the rapidly increasing number of alternatives, it would be infeasible to obtain a reasonably sized input for constraint solvers without pruning clique candidates.

Therefore, we present a number of principles that can be used in practice to cut down the size of the clique database. A labeling L of a chordal graph is *proper* iff L satisfies the conditions of Definition 3. Thus, given a decomposable labeled graph G_L, the labeling L is proper by definition.

Lemma 1 *Consider a chordal graph G and two different sets L and L^* of proper labelings. Let E^c_L and $E^c_{L^*}$ denote the set of edges of a maximal clique c labeled by L and L^*, respectively. If $E^c_L \subseteq E^c_{L^*}$ and $v(c, L) > v(c, L^*)$, then $v(c, L^*)$ is irrelevant for model optimality.*

Proof Since L and L^* are both proper and $E^c_L \subseteq E^c_{L^*}$, any restrictions to the set of labeled edges in c satisfied by L^* will automatically be satisfied by L. Further, as $v(c, L) > v(c, L^*)$, c cannot be labeled by L^* in an optimal labeled graph and thus the labeling L^* and its score $v(c, L^*)$ can be omitted. □

A consequence of Lemma 1 is that for any maximal clique $c \in C$ and any proper labeling L^* such that $v(c, \emptyset) > v(c, L^*)$, the clique c cannot be labeled by L^* in an optimal labeled graph and L^* need not be considered. Thus the main purpose of Lemma 1 is to prune the collection of cliques and the related information, i.e., scores and labels, before applying constraint-based search methods. In the following, we sometimes use the abbreviation $v(c) = v(c, \emptyset)$.

There are, however, further criteria that can be used to filter clique candidates more aggressively. The downside of applying such filtering methods is that too aggressive pruning may sacrifice optimal models. It is worth noting that Bayesian score-based pruning of models as part of the search was proposed in Madigan and Raftery (1994) already. Our first pruning principle is based on *Bayes factors* (Kass and Raftery 1995) and can be used to identify edges that are weakly supported (or unsupported) by the data. Applying the scoring functions defined in the context of the log marginal likelihood (3), we provide the following definition. For later reference, we introduce the abbreviation *bforig* standing for the original Bayes factor.

Definition 4 (*bforig*) Let $\{X, Y\}$ be a clique of two random variables. The mutual dependence of X and Y is weakly supported by the data $\mathbf{X}$ if the log Bayes factor

$$\mathrm{bf}(X, Y) = v(\{X, Y\}) - v(\{X\}) - v(\{Y\}) < \log(10^{-k}), \quad (4)$$

where $k = 0, 1, 2, \ldots$ is a parameter.

It is worth pointing out that (4) does not mention the labeling L since two-element cliques cannot have labels, making the scores independent of L. The condition can be used to remove any clique candidate c such that $\{X, Y\} \subseteq c$. Moreover, the parameter $k = 0, 1, 2, \ldots$ controls the depth of filtering: the higher the value k takes, the fewer cliques will be pruned. However, as far as any larger clique candidates c are considered, condition (4) does not properly take the context created by c into account. This suggests a generalization of the log Bayes factor $\mathrm{bf}(X, Y)$ in the context of c.

Definition 5 *(Bayes factor)* Given a clique c and variables $X, Y \in c$, define

$$\mathrm{bf}(X, Y \mid c) = v(c) - v(c \setminus \{X\}) - v(c \setminus \{Y\}) + v(c \setminus \{X, Y\}). \quad (5)$$

Definition 5 allows us to generalize condition (4) for larger cliques while omitting labelings, which effectively amounts to assuming an empty labeling $\emptyset$. Depending on the point of interest, we introduce the abbreviations *node*, *part*, and *edge* for the following three generalizations.

Definition 6 (*node*, *part*, *edge*) A clique c is weakly supported by the data $\mathbf{X}$ if one of the following conditions holds:

1. *node*: There is a variable $X \in c$ such that

$$\sum_{Y \in c \setminus \{X\}} \frac{\mathrm{bf}(X, Y \mid c)}{|c| - 1} < \log(10^{-k}). \quad (6)$$

2. *part*: There is a partitioning of c into $c_1 \sqcup c_2$ such that

$$\sum_{X \in c_1, Y \in c_2} \frac{\mathrm{bf}(X, Y \mid c)}{|c_1||c_2|} < \log(10^{-k}). \quad (7)$$

3. *edge*: There are two variables $X, Y \in c$ such that

$$\mathrm{bf}(X, Y \mid c) < \log(10^{-k}). \tag{8}$$

By setting $c = \{X, Y\}$, $c_1 = \{X\}$, and $c_2 = \{Y\}$, the conditions (6)–(8) coincide with (4) but their intended use is different. The principle of Definition 4 applies to any clique containing a suspicious edge, whereas the ones of Definition 6 are clearly clique-specific criteria. Since these criteria aim at checking the consistency of scores at edge level, factors (6) and (7) are averaged by the respective numbers of edges $|c| - 1$ and $|c_1||c_2|$ involved. These nominators have no effect when $k = 0$ and $\log(10^{-k}) = 0$, but they regulate the effect of pruning when $k > 0$ grows: the higher the value of k, the fewer cliques will be pruned.

Proposition 1 *Given a clique candidate c, condition* (6) *implies condition* (7) *that implies condition* (8).

Proof The first implication is clear by setting $c_1 = \{X\}$ and $c_2 = c \setminus \{X\}$. For the second implication, suppose that (7) holds for $c = c_1 \sqcup c_2$ but for each $X, Y \in c$, $\mathrm{bf}(X, Y \mid c) \geq \log(10^{-k})$. This contradicts (7), since the sum of log Bayes factors $\sum_{X \in c_1, Y \in c_2} \mathrm{bf}(X, Y \mid c) \geq |c_1||c_2| \log(10^{-k})$. □

Under the assumption that the data generating distribution is *faithful* (Koller and Friedman 2009) to a chordal graph, the conditions defined in (4) and (6)–(8) will not prune essential cliques when the size of the data goes to infinity. Here, a distribution is said to be faithful to a graph if any marginal or conditional independence present in the distribution can be determined from the graph. In such a scenario, as the size of the data goes to infinity, we can assume that an optimal graph, which maximizes the marginal likelihood, will correctly convey all conditional and marginal independencies in the generating distribution. We can further assume that implementations (4) and (5) of the Bayes factor will perform consistently. Consider a clique c in an optimal graph and any pair $\{X, Y\} \subseteq c$. As there is an edge between X and Y, we know that they are conditionally dependent in the generating distribution given any other set of variables, including the empty set and $c \setminus \{X, Y\}$. This means that the functions $\mathrm{bf}(\ldots)$, defined in (4) and (5), will tend to infinity as the size of data tends to infinity. Subsequently, the essential clique c will not be pruned.

In this paper, the experiments are performed on three datasets concerning heart disease (6 binary variables), economical behavior (8 binary variables), or election behavior (25 binary variables), respectively. See, e.g., (Nyman et al. 2014; Pensar et al. 2015) for further details on these datasets abbreviated by *heart*, *econ*, and *hs* in the sequel. The sets *heart* and *econ* are based on complete clique databases, whereas that of *hs* contains all candidates up to size 8. To illustrate the effect of filtering, the column *all* of Table 2 gives the numbers of cliques involved in the datasets. In the labeled case, potential labelings have already been pruned using Lemma 1. The original numbers of candidates for *heart-lab* and *econ-lab* were 506 and 2, 038, respectively. The mnemonics of the last four columns refer back to Definitions 4 and 6. The filtering scheme *bforig* is based on the *deletion* of cliques c containing an edge $\{X, Y\}$ such that condition (4) holds. The filtering schemes *node*, *part*, and *edge* amount to the three conditions (6)–(8) for clique removal, respectively. Each scheme is implemented so that the resulting clique database remains *downward closed*, i.e., if a particular clique c is present in the database so are its all proper subcliques. This is essential for scoring based on perfect elimination orderings to be introduced in the next section. Summarizing Table 2, the *edge* scheme provides the greatest pruning effect, having the negative effect of an increased risk of sacrificing optimal models (cf. Table 7). In the labeled case, the relationships between the pruning criteria are somewhat blurred by the nonuniform distribution of different labelings over clique candidates, which are nevertheless filtered in the same way as without labels.

We consider filtering as off-line computation that is performed only once. In fact, the implementation was not designed to be particularly efficient. For *econ* and *heart*, filtering takes only fractions of a second and is thus negligible. In case of *hs*, the number of cliques subject to filtering is essentially higher. When $k = 0$, the filtering times are 211, 501, 8833, and 299 seconds for the schemes *bforig*, *node*, *part*, and *edge*, respectively. The times do not vary substantially when $k = 1$ and $k = 2$. The considerably longer filtering time for the *part* scheme is due to the high number of cliques as well as the variety in which each clique can be partitioned.

4 Perfect elimination orderings

Undirected trees have the following recursive property: if a leaf node and the respective edge are eliminated, the result is still a tree and the entire tree can be eliminated in this way. This gives rise to an elimination ordering for the nodes of the tree. Since chordal graphs generalize undirected trees, the same idea can be applied to chordal graphs (Chandran et al. 2003; Galinier et al. 1995).

Definition 7 (*Perfect elimination ordering*) Let $G = \langle N, E \rangle$ be an undirected graph. Further, let $X_1 < \cdots < X_n$ be a strict total ordering of N. Define $N_>(X)$ as the set of neighbors of X that follow X in the ordering. Then, $<$ is a *perfect elimination ordering* (PEO) for G if $N_>(X)$ is a clique of G for every $X \in N$.

Perfect elimination orderings enable a chordality test.

Table 2 The effects of different filtering schemes based on the number of remaining clique candidates for given sets of conditions

Dataset	$\|N\|$	k	*all*	*bforig*	*node*	*part*	*edge*
heart	6	0	63	20	18	18	17
		1		39	32	32	24
		2		63	44	44	41
heart-lab	6	0	156	41	34	34	31
		1		95	69	73	47
		2		156	108	108	96
econ	8	0	255	57	115	88	41
		1		191	139	115	64
		2		255	160	141	97
econ-lab	8	0	512	140	331	234	86
		1		432	416	329	152
		2		512	460	424	253
hs	25	0	1,807,780	5818	11,792	7968	1235
		1		188,689	17,727	12,896	2399
		2		1,807,780	26,101	20,202	4713

Theorem 1 (Dirac 1961; Rose 1970) *A graph is chordal iff it has a perfect elimination ordering.*

This connection of chordality and PEOs leads to a procedure in which nodes of a graph are eliminated one by one. Any node in the remaining graph that is part of a clique (consisting of one or more nodes) and is not adjacent to nodes outside the clique is eligible for elimination. If all nodes can be eliminated, then the original graph is chordal. Note that the above definition of PEO is equivalent with the standard definition of perfect ordering (Lauritzen 1996), however, below we utilize PEOs also for labeled graphs which are not considered in Lauritzen (1996).

Example 1 Figure 2 illustrates the application of the above procedure to an 8-node Markov network (a chordal graph having random variables as its nodes) according to the PEO $A < C < H < F < B < E < G < D$. Since all nodes are eliminated in the end, the graph is chordal by Theorem 1.

Many other orderings are applicable as well. It is also possible to eliminate nodes in parallel. For example A, C, H, and F could all be eliminated first, and then the clique $\{B, D, E, G\}$ can be eliminated node by node in four steps. □

To enable the scoring of Markov networks, the expression (3) of log marginal likelihood can be reformulated to better fit the purposes of constraint-based optimization.

Proposition 2 *Let $X_1 < \ldots < X_n$ be a PEO for a labeled decomposable graph G_L.*

If $s_i = N_>(X_i)$ and $c_i = s_i \cup \{X_i\}$ are the cliques of G_L induced by $X_1 < \ldots < X_n$ for $1 \leq i \leq n$, and $v(c_i, L)$ is defined as $v(c_i, \emptyset)$ for non-maximal cliques c_i, then the log marginal likelihood of a dataset $\boldsymbol{X}$ given G_L is

$$\log P(\boldsymbol{X} \mid G_L) = \sum_{i=1}^{n} v(c_i, L) - \sum_{i=1}^{n} v(s_i, L).$$

The score differences $d(c_i, X_i, L) = v(c_i, L) - v(s_i, L)$ for $1 \leq i \leq n$ thus enable a differential calculation

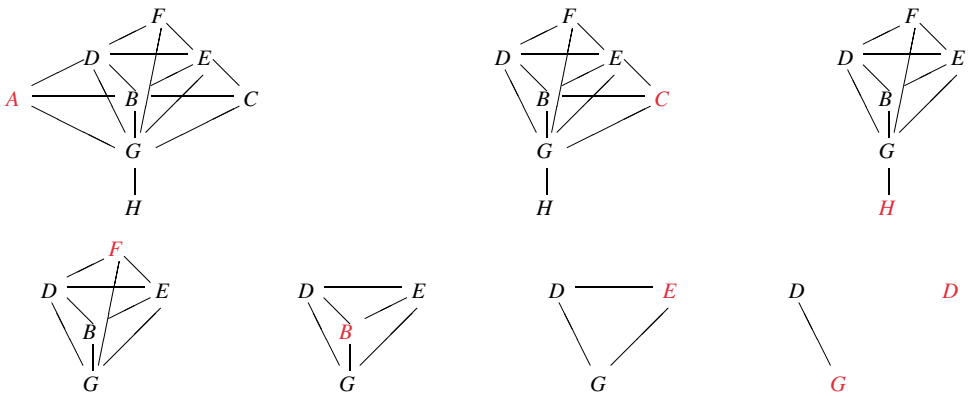


Fig. 2 Applying a perfect elimination ordering to a chordal graph

$$\log P(\mathbf{X} \mid G_L) = \sum_{i=1}^{n} d(c_i, X_i, L). \tag{9}$$

The preceding scheme can be further adjusted by isolating the effects of labelings L on scores. In the following, we let $d(c, X_i) = v(c) - v(c \setminus \{X_i\})$, where c is a clique and $X_i \in c$ a variable. In the special case that $c = \{X_i\}$, we have $d(\{X_i\}, X_i) = v(\{X_i\}) - v(\emptyset) = v(\{X_i\})$. The effect of a labeling L on the score of c can be formalized as a *reward*

$$r(c, L) = v(c, L) - v(c, \emptyset) \tag{10}$$

that is independent on the nodes of c and guaranteed to be positive by Lemma 1. Thus (9) can be rewritten as

$$\log P(\mathbf{X} \mid G_L) = \sum_{i=1}^{n} d(c_i, X_i) + \sum_{i=1}^{n} r(c_i, L). \tag{11}$$

5 Learning Markov networks by constraint optimization

Representing the structure learning problem of decomposable Markov networks is feasible in constraint satisfaction frameworks such as Boolean satisfiability (SAT) (Cook 1971) and its extension by optimization capabilities, i.e., maximum satisfiability (MAXSAT) (Johnson 1974). Also other extensions such as SAT modulo theories (SMT) (Sebastiani and Tomasi 2012) and integer programming (IP) (Gomory 1958) could be used. Answer set programming (ASP) (Brewka et al. 2011; Lifschitz 2002; Marek and Truszczyński 1999; Niemelä 1999) offers similar primitives in the form of rules rather than propositional formulas or linear equations. We begin by reviewing the proof-of-concept representation of the learning problem for Markov networks (Corander et al. 2013). These ideas form the starting point for developing an improved encoding for Markov networks and generalizing the encoding to the *labeled* case.

Definition 8 (*Solutions* (Corander et al. 2013)) Given a set N of nodes representing random variables and a scoring function $v : \mathbf{2}^N \to \mathbb{R}$ based on log marginal likelihoods (3), a set $C = \{c_1, \ldots, c_n\}$ of cliques is a *solution* to the network learning problem iff

1. every node is *covered* by a clique, i.e., $\bigcup_{i=1}^{n} c_i = N$,
2. cliques in C are set-inclusion *maximal*,
3. the graph $\langle N, E\rangle$ with $E = \bigcup_{c \in C} E_c$ is *chordal*,
4. the set C has a *maximum weight spanning tree* labeled by a set $S = \{s_1, \ldots, s_m\}$ of *separators*, and
5. C and S maximize $v(C, S) = \sum_{c \in C} v(c) - \sum_{s \in S} v(s)$.

The constraints on solutions given in Definition 8 can be encoded in any of the mentioned formalisms for constraint optimization. For instance, if a Boolean (0-1) variable in_c represents that a clique c is a part of a candidate solution, the first constraint can be formalized by a disjunction $\mathrm{in}_{c_1} \vee \ldots \vee \mathrm{in}_{c_k}$, where $c_1, \ldots, c_k$ are all cliques that contain a particular node. Using dedicated variables $\mathrm{e}_{i,j}$ to represent the edges of the resulting graph, expressing the maximality of cliques is straightforward. In contrast, obtaining a compact representation for chordality and the existence of a maximum weight spanning tree gets far more involved. For small graphs, however, chordality could be enforced by explicitly denying chordless cycles of length four or more. For the spanning tree condition, a key idea of Corander et al. (2013) is that the maximum weight of separators can be replaced by a *balancing condition*: every node occurs in the chosen cliques exactly once more than in the respective separators. This enables the recognition of separators in terms of standard *cardinality constraints* (Sinz 2005) and hence allows for determining the overall score (3) of the resulting network.

5.1 Encoding based on PEOs

In what follows, we present an encoding of the Markov network learning problem that improves the one summarized above in a number of ways. In particular, the idea is to exploit PEOs in the encoding of the chordality check and the results of Sect. 4 for scoring Markov networks. The goal is to avoid the determination of separators as far as possible. This is because the connection of a separator to the cliques it separates from each other gives rise to a cubic relation in the number of candidate cliques, i.e., substantial space complexity. We thus restate the requirements from Definition 8.

Definition 9 (*Solutions reformulated*) Given a function $d : \mathbf{2}^N \times N \to \mathbb{R}$, a graph $\langle N, E\rangle$ based on $N = \{X_1, \ldots, X_n\}$ is a solution to the Markov network learning problem iff

1. every node X_i has a clique c_i as its *context* of *elimination*, where $X_i \in c_i$ and $E = \bigcup_{i=1}^{n} c_i$,
2. there is a perfect elimination ordering for $\langle N, E\rangle$, and
3. $\sum_{i=1}^{n} d(c_i, X_i)$ corresponding to the log marginal likelihood (9) is maximized.

The elimination of X_i in the context of c_i means that X_i and all edges of c_i incident with X_i are removed. For instance, the node A is removed in the context of $\{A, B, D, G\}$ in Fig. 2. To select a candidate graph, we introduce a Boolean variable in_i^c for each node X_i and clique candidate c with $X_i \in c$. If $c_1, \ldots, c_m$ are the clique candidates containing X_i, the choice of the elimination context for X_i can be expressed by $\mathrm{in}_i^{c_1} \vee \ldots \vee \mathrm{in}_i^{c_m}$. This disjunction should be made exclusive by adding $\neg\mathrm{in}_i^{c_j} \vee \neg\mathrm{in}_i^{c_k}$ for all $1 \leq j < k \leq m$, or by

Table 3 Boolean variables used in the PEO-based encoding

Variable	Intuitive reading
in_i^c	Variable X_i is eliminated in context c
$\text{el}_{i,j}$	Variable X_i is eliminated at step j
$\text{el}_{i,j}^c$	Variable X_i is eliminated before step j or off context c
cr_i^c	Variable X_i creates edges for context c
$\text{sf}_{i,j}$	The j first score fragments of X_i count

adding an equivalent cardinality constraint allowing only one of the variables $\text{in}_i^{c_1}, \ldots, \text{in}_i^{c_m}$ to be true for X_i. The Boolean variables utilized by the encoding are collected in Table 3 for the reader's convenience.

Our next objective is to enforce the chordality of the selected graph. This will be achieved by checking the existence of a PEO for the graph as formalized by Theorem 1. However, rather than insisting on a total ordering of the nodes, we allow for parallel eliminations to limit the number of steps and to achieve a more compact encoding. If parallel eliminations are considered, the worst case can be illustrated by an n-element clique $\{X_1, X_2, \ldots, X_n\}$ whose elimination requires n elimination steps using some arbitrary ordering as PEO (cf. the clique $\{B, D, E, G\}$ in Fig. 2). The reason is that the potential removals of B, D, E, and G compete over the same resources (edges) and cannot be done at once in view of the differential score calculation (9).

To formalize the chordality test for a variable X_i, we introduce Boolean variables $\text{el}_{i,j}$ and $\text{el}_{i,j}^c$ for each elimination step $1 \leq j \leq n$ and clique candidate c such that $X_i \in c$. The role of $\text{el}_{i,j}^c$ is to signal the elimination of X_i in context c at step $j-1$ to the elements of $c \setminus \{X_i\}$, which may then be eliminated from step j on. In turn, let $c_1, \ldots, c_m$ be all possible elimination contexts for variables $X_{i_1} \in c_1, \ldots, X_{i_m} \in c_m$ such that $X_i \in c_1 \setminus \{X_{i_1}\}, \ldots, X_i \in c_m \setminus \{X_{i_m}\}$. Then, X_i can be eliminated once all X_{i_k} for $1 \leq k \leq m$ whose elimination context is c_k, as indicated by $\text{in}_{i_k}^{c_k}$, are. This condition is captured by

$$\text{el}_{i,j} \leftrightarrow \text{el}_{i_1,j}^{c_1} \wedge \ldots \wedge \text{el}_{i_m,j}^{c_m}, \tag{12}$$

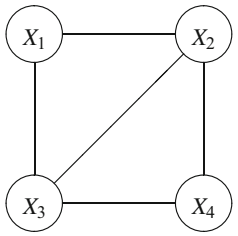
where the idea that $\text{el}_{i_k,j}^{c_k}$ is true if X_{i_k} is eliminated at some earlier step than j or in another context than c_k is recursively formalized by

$$\text{el}_{i_k,j}^{c_k} \leftrightarrow \text{el}_{i_k,j-1} \vee \neg\text{in}_{i_k}^{c_k} \tag{13}$$

for $j > 1$ as well as $\text{el}_{i_k,1}^{c_k} \leftrightarrow \neg\text{in}_{i_k}^{c_k}$ for the base case $j = 1$. The definitions above aim at expressing chordality as

$$\text{el}_{1,n} \wedge \ldots \wedge \text{el}_{n,n},$$

Fig. 3 Illustration of the encoding of the chordality test



requiring the elimination of all nodes in the end. Due to recursion over elimination steps, the overall space complexity is quadratic in n. However, depending on the target formalism, a linear representation can be feasible. This holds, e.g., for encodings in ASP that natively support recursion.

Example 2 To illustrate the encoding of the chordality condition, let us consider the Markov network in Fig. 3 and the following candidates:

$$\begin{aligned} c_1 &= \{X_1\}, & c_2 &= \{X_2\}, \\ c_3 &= \{X_3\}, & c_4 &= \{X_4\}, \\ c_5 &= \{X_1, X_2\}, & c_7 &= \{X_2, X_3\}, \\ c_6 &= \{X_1, X_3\}, & c_8 &= \{X_2, X_4\}, \\ & & c_9 &= \{X_3, X_4\}, \\ c_{10} &= \{X_1, X_2, X_3\}, & c_{11} &= \{X_2, X_3, X_4\}. \end{aligned}$$

Since X_1 is mentioned by cliques c_1, c_5, c_6, and c_{10}, the choice of an elimination context is expressed by

$$\begin{aligned} &\text{in}_1^{c_1} \vee \text{in}_1^{c_5} \vee \text{in}_1^{c_6} \vee \text{in}_1^{c_{10}}, \\ &\neg\text{in}_1^{c_1} \vee \neg\text{in}_1^{c_5}, \quad \neg\text{in}_1^{c_1} \vee \neg\text{in}_1^{c_6}, \quad \neg\text{in}_1^{c_1} \vee \neg\text{in}_1^{c_{10}}, \\ &\neg\text{in}_1^{c_5} \vee \neg\text{in}_1^{c_6}, \quad \neg\text{in}_1^{c_5} \vee \neg\text{in}_1^{c_{10}}, \quad \neg\text{in}_1^{c_6} \vee \neg\text{in}_1^{c_{10}}. \end{aligned}$$

Analogous formulas are needed for the nodes X_2, X_3, and X_4. Let us then consider a concrete elimination scenario, where X_1 and X_4 are first removed in parallel, then X_2 is removed, and finally X_3. This scenario can be realized by setting the variables $\text{in}_1^{c_{10}}$, $\text{in}_2^{c_7}$, $\text{in}_3^{c_3}$, and $\text{in}_4^{c_{11}}$ true. The resulting order of elimination is depicted in Fig. 4a. Since $\text{in}_1^{c_{10}}$ is true, the other variables $\text{in}_1^{c_1}$, $\text{in}_1^{c_5}$, and $\text{in}_1^{c_6}$ concerning X_1 are falsified by mutual exclusion. Respective $\text{in}_i^{c_j}$-variables related with X_2, X_3, and X_4 get similarly falsified. Let us then consider the satisfaction of formulas of forms (12) and (13). For X_1 and the elimination step $j = 1$, the relevant instances are

$$\begin{aligned} &\text{el}_{1,1} \leftrightarrow \text{el}_{2,1}^{c_5} \wedge \text{el}_{3,1}^{c_6} \wedge \text{el}_{2,1}^{c_{10}} \wedge \text{el}_{3,1}^{c_{10}}, \\ &\text{el}_{2,1}^{c_5} \leftrightarrow \neg\text{in}_2^{c_5}, \quad \text{el}_{3,1}^{c_6} \leftrightarrow \neg\text{in}_3^{c_6}, \\ &\text{el}_{2,1}^{c_{10}} \leftrightarrow \neg\text{in}_2^{c_{10}}, \quad \text{el}_{3,1}^{c_{10}} \leftrightarrow \neg\text{in}_3^{c_{10}}. \end{aligned}$$

Due to mutual exclusions explained above, the variables $\text{el}_{2,1}^{c_5}$, $\text{el}_{3,1}^{c_6}$, $\text{el}_{2,1}^{c_{10}}$, and $\text{el}_{3,1}^{c_{10}}$ must be set to true. This, in turn, makes $\text{el}_{1,1}$ true, indicating that X_1 is removed at step $j = 1$. Since X_4 is symmetric to X_1, we may conclude that $\text{el}_{4,1}$ is also

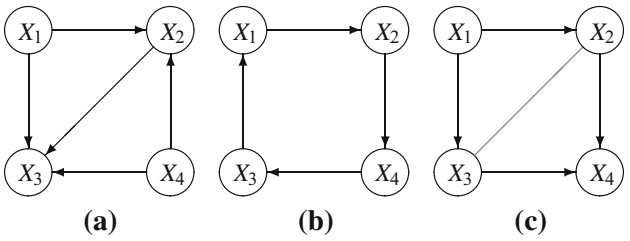


Fig. 4 Illustration of elimination contexts and the resulting orderings of nodes

made true by analogous equivalences introduced for X_4. On the other hand, the respective formulas for X_2 and X_3 falsify both $\mathrm{el}_{2,1}$ and $\mathrm{el}_{3,1}$ since both $\mathrm{in}_1^{c_{10}}$ and $\mathrm{in}_4^{c_{11}}$ are true. Thus X_2 and X_3 are not eliminated at step $j = 1$. As regards the subsequent elimination of X_2 at step $j = 2$, the following instances of formulas (12) and (13) become relevant:

$$\begin{aligned}
&\mathrm{el}_{2,2} \leftrightarrow \mathrm{el}_{1,2}^{c_5} \wedge \mathrm{el}_{3,2}^{c_7} \wedge \mathrm{el}_{4,2}^{c_8} \wedge \mathrm{el}_{1,2}^{c_{10}} \wedge \mathrm{el}_{3,2}^{c_{10}} \wedge \mathrm{el}_{3,2}^{c_{11}} \wedge \mathrm{el}_{4,2}^{c_{11}},\\
&\mathrm{el}_{1,2}^{c_5} \leftrightarrow \mathrm{el}_{1,1} \vee \neg\mathrm{in}_1^{c_5}, \qquad \mathrm{el}_{3,2}^{c_7} \leftrightarrow \mathrm{el}_{3,1} \vee \neg\mathrm{in}_3^{c_7},\\
&\mathrm{el}_{4,2}^{c_8} \leftrightarrow \mathrm{el}_{4,1} \vee \neg\mathrm{in}_4^{c_8},\\
&\mathrm{el}_{1,2}^{c_{10}} \leftrightarrow \mathrm{el}_{1,1} \vee \neg\mathrm{in}_1^{c_{10}}, \qquad \mathrm{el}_{3,2}^{c_{10}} \leftrightarrow \mathrm{el}_{3,1} \vee \neg\mathrm{in}_3^{c_{10}},\\
&\mathrm{el}_{3,2}^{c_{11}} \leftrightarrow \mathrm{el}_{3,1} \vee \neg\mathrm{in}_3^{c_{11}}, \qquad \mathrm{el}_{4,2}^{c_{11}} \leftrightarrow \mathrm{el}_{4,1} \vee \neg\mathrm{in}_4^{c_{11}}.
\end{aligned}$$

To check the satisfaction of the conjunction in the equivalence for $\mathrm{el}_{2,2}$, it is sufficient to note that $\mathrm{el}_{1,1}$ and $\mathrm{el}_{4,1}$ have been set to true before, and the truth of $\mathrm{in}_3^{c_7}$, $\mathrm{in}_3^{c_{10}}$, and $\mathrm{in}_3^{c_{11}}$ is excluded by the truth of $\mathrm{in}_3^{c_3}$. Thus $\mathrm{el}_{2,2}$ must be true. The analogous formulas for X_1 and X_4 force $\mathrm{el}_{1,2}$ and $\mathrm{el}_{4,2}$ to be true. To the contrary, $\mathrm{el}_{3,2}$ is falsified since $\mathrm{in}_2^{c_7}$ is true and $\mathrm{el}_{2,1}$ is false. Our last observations concern the elimination step $j = 3$, where X_3 becomes eligible for elimination in the context of c_3. Since the equivalence for $\mathrm{el}_{3,3}$ depends positively on $\mathrm{el}_{1,2}$, $\mathrm{el}_{2,2}$, and $\mathrm{el}_{4,2}$ only, it is clear that $\mathrm{el}_{3,3}$ is set to true. The same can be stated about the other nodes, i.e., $\mathrm{el}_{1,3}$, $\mathrm{el}_{2,3}$, and $\mathrm{el}_{4,3}$ are made true by the respective equivalences. This illustrates the persistence of elimination: if X_i is eliminated at step j, it will remain eliminated at step $j + 1$. In particular, the formula $\mathrm{el}_{1,3} \wedge \mathrm{el}_{2,3} \wedge \mathrm{el}_{3,3} \wedge \mathrm{el}_{4,3}$ is true as an indication of chordality in the scenario discussed so far.

In order to demonstrate further aspects of the chordality encoding, two other scenarios deserve attention. First, let us assume that $\mathrm{in}_1^{c_5}$, $\mathrm{in}_2^{c_8}$, $\mathrm{in}_3^{c_6}$, and $\mathrm{in}_4^{c_9}$ have been chosen to be true. This creates an interdependency for the selected elimination contexts, so that no node can be eliminated as illustrated in Fig. 4b. On the logical side this implies that $\mathrm{el}_{i,j}$ will be falsified for each X_i and $j = 1, \ldots, 4$. Indeed, a graph in which the diagonal edge from Fig. 3 has been removed is not chordal, also witnessed by the falsity of $\mathrm{el}_{1,4} \wedge \mathrm{el}_{2,4} \wedge \mathrm{el}_{3,4} \wedge \mathrm{el}_{4,4}$.

Second, let us pick $\mathrm{in}_1^{c_{10}}$, $\mathrm{in}_2^{c_8}$, $\mathrm{in}_3^{c_9}$, and $\mathrm{in}_4^{c_4}$ to be true. This represents a scenario in which X_1 is removed at step $j = 1$ with $c_{10} = \{X_1, X_2, X_3\}$ as its elimination context. This suggests that the edge $\{X_2, X_3\}$ is present in the graph but, unfortunately, the elimination contexts of X_2 and X_3 do not include this edge, colored gray in Fig. 4c. The formulas of the chordality encoding will still set the variables $\mathrm{el}_{1,1}$, $\mathrm{el}_{2,2}$, $\mathrm{el}_{3,2}$, and $\mathrm{el}_{4,3}$ to true. Therefore, we observe that further constraints are necessary to ensure that the structure claimed to exist by the chosen elimination contexts of the nodes is indeed present in the graph. □

A new Boolean variable cr_i^c captures the idea that node X_i is responsible for the *creation* of the structure present in the clique c when elimination contexts are initially chosen. The constraint we want to impose on the elimination context c of a node X_i is essentially

$$\mathrm{in}_i^c \rightarrow \mathrm{cr}_{i_1}^{c'} \vee \ldots \vee \mathrm{cr}_{i_k}^{c'}, \tag{14}$$

where $c' = c \setminus \{X_i\} = \{X_{i_1}, \ldots, X_{i_k}\}$ gives the remainder of c if X_i is eliminated in this context. It remains to provide a formula defining the truth value of cr_i^c:

$$\mathrm{cr}_i^c \leftrightarrow \mathrm{in}_i^c \vee \mathrm{cr}_i^{c_1} \vee \ldots \vee \mathrm{cr}_i^{c_k}, \tag{15}$$

where $c_1 = c \cup \{X_1\}, \ldots, c_k = c \cup \{X_k\}$ are all clique candidates extending c by one node. The effect of these formulas is illustrated next.

Example 3 Let us continue from the last scenario in Example 2, i.e., Fig. 4c. The relevant instance of (14) is $\mathrm{in}_1^{c_{10}} \rightarrow \mathrm{cr}_2^{c_7} \vee \mathrm{cr}_3^{c_7}$, implying the consequent $\mathrm{cr}_2^{c_7} \vee \mathrm{cr}_3^{c_7}$. Thus either X_2 or X_3 is responsible for creating the structure present in c_7. To check this, we need to evaluate the respective instances of (15):

$$\begin{aligned}
&\mathrm{cr}_2^{c_7} \leftrightarrow \mathrm{in}_2^{c_7} \vee \mathrm{cr}_2^{c_{10}} \vee \mathrm{cr}_2^{c_{11}},\\
&\mathrm{cr}_3^{c_7} \leftrightarrow \mathrm{in}_3^{c_7} \vee \mathrm{cr}_3^{c_{10}} \vee \mathrm{cr}_3^{c_{11}},\\
&\mathrm{cr}_2^{c_{10}} \leftrightarrow \mathrm{in}_2^{c_{10}}, \qquad \mathrm{cr}_2^{c_{11}} \leftrightarrow \mathrm{in}_2^{c_{11}},\\
&\mathrm{cr}_3^{c_{10}} \leftrightarrow \mathrm{in}_3^{c_{10}}, \qquad \mathrm{cr}_3^{c_{11}} \leftrightarrow \mathrm{in}_3^{c_{11}}.
\end{aligned}$$

But since $\mathrm{in}_2^{c_7}$, $\mathrm{in}_3^{c_7}$, $\mathrm{in}_2^{c_{10}}$, $\mathrm{in}_3^{c_{10}}$, $\mathrm{in}_2^{c_{11}}$, and $\mathrm{in}_3^{c_{11}}$ have been set to false, the variables $\mathrm{cr}_2^{c_7}$, $\mathrm{cr}_3^{c_7}$, $\mathrm{cr}_2^{c_{10}}$, $\mathrm{cr}_3^{c_{10}}$, $\mathrm{cr}_2^{c_{11}}$, and $\mathrm{cr}_3^{c_{11}}$ must be falsified in turn. This rules out the possibility of satisfying $\mathrm{cr}_2^{c_7} \vee \mathrm{cr}_3^{c_7}$ inferred above, i.e., the scenario under consideration is no longer possible. □

Our last requirement for solutions to the Markov network learning problem concerns the score of a chordal graph for which a PEO can be identified. The idea is to use a differential score $d(c, X_i)$ based on (9) that is compatible with X_i being eliminated in the context of c. If $c_1, \ldots, c_m$ is the collection

of candidate cliques, the resulting *objective function* can be written as $\sum_{j=1}^{m} \sum_{X_i \in c_j} d(c_j, X_i)\mathrm{in}_i^{c_j}$.

The representation of the objective function can be modularized by taking the variable view. So, let X_i be one of the variables with $1 \leq i \leq n$ and $c_1, \ldots, c_k$ the cliques containing X_i. Since clique candidates are known in advance, we may assume that $d(c_1, X_i) \geq \ldots \geq d(c_k, X_i)$ without loss of generality. Because X_i must have an elimination context, it is clear that X_i will be assigned at least the first score $d(c_1, X_i)$ and potentially some smaller *score fractions* expressible as $d(c_j, X_i) - d(c_{j-1}, X_i)$ for $1 < j \leq k$. To control which fractions are needed, we introduce Boolean variables $\mathrm{sf}_{i,j}$ for each clique c_j with $1 < j < k$. The activation of score fractions is determined by the formula

$$\mathrm{sf}_{i,j} \leftrightarrow \mathrm{in}_i^{c_j} \vee \mathrm{sf_{i,j+1}}, \tag{16}$$

where $1 < j < k$ refers to an elimination context c_j of X_i. The base case $j = k$, is covered by the formula

$$\mathrm{sf}_{i,k} \leftrightarrow \mathrm{in}_i^{c_k}. \tag{17}$$

The idea is that, if $\mathrm{sf}_{i,j}$ is set to true, then all other variables $\mathrm{sf}_{i,j'}$ with $j' < j$ are set to true as well. In this way, the contribution of X_i to the objective function is

$$o(i) = d(c_1, X_i) + \sum_{j=2}^{k} (d(c_j, X_i) - d(c_{j-1}, X_i))\mathrm{sf}_{i,j}. \tag{18}$$

The sum $o(i)$ equals to $d(c_j, X_i)$ for the chosen elimination context c_j of X_i. The goal of this encoding is to allow smooth changes of scores when the elimination contexts of X_i are switched in accordance with the order $c_1, \ldots, c_k$.

Theorem 2 *Let $N = \{X_1, \ldots, X_n\}$ be a set of random variables and $d : \mathbf{2}^N \times N \to \mathbb{R}$ a scoring function. An undirected graph $\langle N, E\rangle$ is a solution to the Markov network learning problem iff*

1. *the elimination formulas* (12) *are satisfied for each X_i and step $1 \leq j \leq n$ together with all instances of* (13)*,*
2. *the formula $\mathrm{el}_{1,n} \wedge \ldots \wedge \mathrm{el}_{n,n}$ is satisfied,*
3. *the context creation formulas* (14) *and* (15) *are satisfied for each X_i and each clique c such that $X_i \in c$, and*
4. *the graph $\langle N, E\rangle$ maximizes the sum $\sum_{i=1}^{n} o(i)$ of objectives* (18)*, while satisfying all formulas* (16) *and* (17) *defining score fractions.*

Proof sketch We argue that a graph $\langle N, E\rangle$ based on N is chordal iff the requirements 1–3 are met.

($\Longrightarrow$) Let $\langle N, E\rangle$ be a chordal graph. By Theorem 1, the underlying graph $\langle N, E\rangle$ has a PEO based on a strict total ordering of N. This ordering can be relaxed by recursively taking at step j as many as possible variables X_i from the ordering not competing for edges. In this way, the context elimination formulas get satisfied. The context creation formulas are satisfied because every variable X_i subject to elimination in its context c_i is connected to a clique $c_i \setminus \{X_i\}$.

($\Longleftarrow$) Suppose that requirements 1–3 are met by $\langle N, E\rangle$ and let I be the satisfying assignment. A variable X_i is eliminated at step $j \geq 1$ if $\mathrm{el}_{i,j}$ is true in I and each $\mathrm{el}_{i,j'}$ with $1 \leq j' < j$ is false in I. A PEO can be constructed by taking variables eliminated at the same step j in any order. Since I satisfies the context creation formulas, it is guaranteed that, when X_i is eliminated, it is connected to a clique c such that a context $c_i = c \cup \{X_i\}$ of elimination is determined for X_i. Thus $\langle N, E\rangle$ is chordal by Theorem 1.

Finally, we note that objective functions used for scoring, i.e., $\sum_{i=1}^{n} d(c_i, X_i)$ and $\sum_{i=1}^{n} o(i)$, coincide.

Theorem 2 is applicable to any downward closed collection of cliques, where d may be a partial scoring function.

5.2 Encoding labels for Markov networks

The labels representing context-dependent conditional probabilities can be incorporated by an orthogonal extension to the basic encoding. It is essential to ensure that edges subject to a labeling L are not contained in the separators of the underlying chordal graph. Because the encoding developed in Sect. 5.1 avoids the formalization of separators altogether, we have to detect them somehow. For the sake of space efficiency, we concentrate on identifying edges involved in separators rather than identifying separators themselves (Table 4). Thus a Boolean variable $\mathrm{e}_{i,j}$ is introduced for each pair X_i, X_j of nodes with $1 \leq i < j \leq n$, to be set to true if and only if the edge between X_i and X_j is present. If $c_1, \ldots, c_k$ are the cliques containing both X_i and X_j, we obtain the formula

$$\mathrm{e}_{i,j} \leftrightarrow \mathrm{in}_i^{c_1} \vee \mathrm{in}_j^{c_1} \vee \ldots \vee \mathrm{in}_i^{c_k} \vee \mathrm{in}_j^{c_k}. \tag{19}$$

When the edge between X_i and X_j is present in a graph, we are interested in other nodes X_k that are connected by an edge to both X_i and X_j. To detect such nodes, we use a Boolean variable $\mathrm{m}_{i,j,k}$ and a formula

$$\mathrm{m}_{i,j,k} \leftrightarrow \mathrm{e}'_{i,k} \wedge \mathrm{e}'_{j,k}, \tag{20}$$

Table 4 Boolean variables used in the encoding of labels

Variable	Intuitive reading
$\mathrm{e}_{i,j}$	The edge between X_i and X_j is present
$\mathrm{m}_{i,j,k}$	Variable X_k connects to both X_i and X_j
$\mathrm{s}_{i,j}$	The edge between X_i and X_j belongs to a separator
xl^c	Clique c is labeled by some superclique
xl_i^c	Labeling L_i is excluded for c

where $e'_{i,k} = e_{i,k}$ (or $e'_{j,k} = e_{j,k}$) if $i < k$ (or $j < k$), while $e'_{i,k} = e_{k,i}$ (or $e'_{j,k} = e_{k,j}$) otherwise. The edge between X_i and X_j belongs to a separator if and only if it is connected—in the sense specified above—to two distinct nodes X_k and X_l that are not connected by an edge. A Boolean variable $s_{i,j}$ is introduced to capture this condition and its truth value is set in terms of

$$s_{i,j} \leftrightarrow \bigvee_{1 \le k < l \le n} m_{i,j,k} \wedge m_{i,j,l} \wedge \neg e_{k,l}. \tag{21}$$

Example 4 Recall the first scenario of Example 2 in which in_1^{c10}, in_2^{c7}, in_3^{c3}, and in_4^{c11} were set to true. The formulas of the form (19) force the variables $e_{1,2}$, $e_{1,3}$, $e_{2,3}$, $e_{2,4}$, and $e_{3,4}$ to be true, whereas other edge variables remain false.

Thus formulas (20) make variables $m_{2,3,1}$ and $m_{2,3,4}$ true. Other such variables are not relevant for our analysis.

The truth assignments reported above ensure that $m_{2,3,1} \wedge m_{2,3,4} \wedge \neg e_{1,4}$ evaluates to true. This sets $s_{2,3}$ to true by the respective instance of formula (21), indicating that the edge between X_2 and X_3 is involved in a separator, i.e., $s = c_{10} \cap c_{11}$. □

It remains to decide the labeling for cliques. As worked out in the end of Section 4, labelings $L_1, \ldots, L_k$ modify the score of a clique c by rewards $r(c, L_1), \ldots, r(c, L_k)$ according to (10), where $r(c, L_1) \ge \ldots \ge r(c, L_k)$ can be assumed without loss of generality and Lemma 1 implies that L_k is the empty labeling $\emptyset$. Since we aim at maximizing the score for c, we should pick the first labeling from the sequence $L_1, \ldots, L_k$ that satisfies the labeling condition. It is worth noting that $L_k = \emptyset$ satisfies the condition trivially, so that eventually some labeling can be found.

To formalize this idea, we introduce a Boolean variable xl^c denoting that c is labeled by some of its supercliques, and Boolean variables xl_i^c denoting that a particular labeling L_i is *excluded* for c. The truth value of the variable xl^c can be determined by the formula

$$xl^c \leftrightarrow \bigvee_{X_k \notin c, X_j \in c \cup \{X_k\}} in_j^{c \cup \{X_k\}}. \tag{22}$$

As regards variables xl_i^c, the following formula is used to set its truth value in the base case $i = 1$, but only if $k > 1$, i.e., L_1 is different from the empty labeling $\emptyset$:

$$xl_1^c \leftrightarrow \left(\bigvee_{X_j \in c} in_j^c \right) \wedge (xl^c \vee \bigvee \{s_{j,k} \mid L_1 \text{ labels } \{X_j, X_k\}\}). \tag{23}$$

When $1 < i < k$, a recursive formula is used:

$$xl_i^c \leftrightarrow xl_{i-1}^c \wedge \left(xl^c \vee \bigvee \{s_{j,k} \mid L_i \text{ labels } \{X_j, X_k\}\} \right). \tag{24}$$

To implement the objective function (11) in the labeled case, we observe that rewarding an admissible label L_i for $1 \le i < k$ is equivalent to penalizing the exclusion of L_i (and all L_j such that $1 \le j < i$) by a negative reward fraction $r(c, L_{i+1}) - r(c, L_i)$. Thus the required extension to the objective function is

$$r(c, L_1) + \sum_{i=1}^{k-1} (r(c, L_{i+1}) - r(c, L_i)) xl_i^c. \tag{25}$$

In the special case $k = 1$, the sum above reduces to $r(c, \emptyset) = v(c, \emptyset) - v(c, \emptyset) = 0$. This means that the given extension is not needed for cliques that only have one possible labeling, i.e., the empty labeling $\emptyset$ acting as the default labeling.

Theorem 2 can be generalized for labeled Markov networks by extending the set of requirements for a graph candidate $\langle N, E \rangle$. First, the edges involved in separators are detected by satisfying all instances of formulas (19)–(21). Once these have been identified, the best possible labelings for the cliques of $\langle N, E \rangle$ can be orthogonally determined by satisfying the formulas (22)–(24). Finally, the resulting rewards (25) are taken into account in the objective function.

6 Experimental evaluation

We have evaluated our constraint-based approach to Markov network structure learning using a number of fully automated off-the-shelf constraint solvers. To this end, we encoded[1] the conditions and optimization measures described in Section 5 in the input language of ASP and used the tool **LP2ACYC** (Gebser et al. 2014) for automatic translation to MAXSAT as well as IP format. Our comparison includes the solvers

- **CLASP** (version 3.1.1) (Gebser et al. 2012),
- **PWBO** (version 2.2) (Martins et al. 2012),
- **SAT4J** (version 2.3.5) (Berre and Parrain 2010), and
- **CPLEX** (version 12.6.0) [126].

As **CLASP** has been originally devised for ASP solving, we also compare it on input in ASP format, without translation to MAXSAT, and below denote this system variant by CL*ASP*. Note that all solvers apply branch-and-bound techniques that successively refine an upper bound on solution quality, while unsatisfiability-based optimization approaches [cf. Manquinho et al. (2009)] turned out ineffective for the Markov network learning problem and are omitted here. The experiments were run sequentially on a Linux machine having 2.70 GHz Intel Xeon E5-4650 CPUs and 256 GB RAM,

[1] http://research.ics.aalto.fi/software/asp/encodings/.

Table 5 Solver runtimes in seconds for learning context-dependent Markov networks from *heart* and *econ* datasets with different filtering schemes

	heart												***econ***											
	$k=0$				$k=1$				$k=2$				$k=0$				$k=1$				$k=2$			
	b	*n*	*p*	*e*	*b*	*n*	*p*	*e*	*b*	*n*	*p*	*e*	*bforig*	*node*	*part*	*edge*	*bforig*	*node*	*part*	*edge*	*bforig*	*node*	*part*	*edge*
CLASP	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**24**	**4**	**0**	**305**	**61**	**19**	**1**	**900**	**131**	**74**	**5**
CLASP	**0**	**0**	**0**	**0**	1	1	1	**0**	7	1	1	1	326	7008	2300	3	50,903	13,920	9014	127	*TO*	26,056	15,783	1672
PWBO	**0**	**0**	**0**	**0**	3	2	2	**0**	45	5	5	2	11,828	*TO*	*TO*	14	*TO*	*TO*	*TO*	3416	*TO*	*TO*	*TO*	59,414
SAT4J	1	1	1	1	4	6	6	1	36	9	8	4	740	10,218	20,430	14	*TO*	50,318	9809	233	*TO*	*TO*	53,838	6101
CPLEX	**0**	**0**	**0**	**0**	3	1	1	**0**	146	6	5	3	282	13,264	1250	5	*TO*	54,860	14,210	86	*TO*	*TO*	51,561	1543
lab																								
CLASP	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**2**	**136**	**14**	**0**	**452**	**286**	**84**	**1**	**1014**	**480**	**331**	**22**
CLASP	**0**	**0**	**0**	**0**	2	1	1	**0**	43	4	5	1	500	14,692	3066	3	44,111	36,138	13,051	135	73,661	50,430	38,638	1976
PWBO	**0**	**0**	**0**	**0**	9	5	5	**0**	257	11	12	3	12,259	*TO*	*TO*	17	*TO*	*TO*	*TO*	2111	*TO*	*TO*	*TO*	67,131
SAT4J	1	1	1	1	13	11	11	1	168	19	17	7	771	41,525	20,183	18	*TO*	*TO*	35,454	255	*TO*	*TO*	*TO*	7587
CPLEX	**0**	**0**	**0**	**0**	58	19	19	**0**	1777	71	71	27	2735	*TO*	55,804	57	*TO*	*TO*	*TO*	2579	*TO*	*TO*	*TO*	60,594

by imposing a time limit of 86,400 seconds (one day) per run. Timeouts are indicated by entries "*TO*" in tables that follow.

Table 5 provides runtimes in seconds on *heart* and *econ* datasets, relative to the filtering schemes *b(forig)*, *n(ode)*, *p(art)*, and *e(dge)* from Sect. 3, parametrized by $k = 0, 1, 2$. For both benchmarks, the upper five rows refer to the unlabeled problem variants, and the lower five rows to the case of labeled Markov networks. Despite of performance gaps between solvers, the runtimes tightly correlate with the number of clique candidates in the input. As shown in Table 2, this number is regulated by the parameter k as well as the filtering scheme, where *bforig* and *edge* tend to prune cliques least or most aggressively, respectively. Notably, the *bforig* scheme along with $k = 2$ yields no pruning at all and thus reflects the performance on unfiltered datasets. Recalling the account of filtering times from Sect. 3, the filtering of clique candidates leads to effective savings in runtime.

Moreover, we note that the performance differences between unlabeled and labeled inputs obtained with the same filtering scheme are rather moderate, given that best labelings can be read off PEOs (cf. Sect. 5.2) to avoid an (additional) model space explosion as reported in Table 1. Comparing the solvers to each other exhibits that **CL*ASP***, run as an ASP solver, performs best on all inputs, leading to the shortest runtimes highlighted in boldface in each column. This advantage is due to the native support of recursion in ASP, which permits a more compact internal problem representation and resulting improvements in search efficiency (in terms of conflicts) by one to two orders of magnitude in comparison to **CLASP** applied to instances in MAXSAT format. However, the performance of the MAXSAT solvers **CLASP**, **PWBO**, and **SAT4J** as well as the IP solver **CPLEX** follows a similar pattern, witnessing a consistent impact of the number of clique candidates. As a reference, note that the MCMC algorithm in Nyman et al. (2014) has been reported to take tens of minutes for converging to an optimum on *heart-lab* and *econ-lab* datasets, still without proving optimality in view of incompleteness of the method.

Table 6 File sizes of solver inputs in different formats

Dataset	ASP	MAXSAT	IP
heart	41 kB	1375 kB	483 kB
Corander et al. (2013)	197 kB	3120 kB	*TO*
econ	300 kB	13,345 kB	3212 kB
Corander et al. (2013)	4300 kB	133,120 kB	*TO*

Regarding the compactness of encodings, the file sizes in Table 6 give an account of the progress relative to Corander et al. (2013), where the unlabeled *heart* and *econ* datasets have also been investigated. Hence, comparing the rows for either benchmark shows a considerable size reduction due to exploiting PEOs rather than maximal cliques and the balancing condition for scoring. In particular, the space requirements are decreased by one order of magnitude on the *econ* dataset, for both ASP and MAXSAT format, and the IP column further quantifies space savings in comparison to the translation from ASP to MAXSAT. Moreover, Corander et al. (2013) reported a shortest solver runtime of three days for the *econ* dataset, while **CL*ASP*** is now able to find and verify an optimal model within 15 minutes (cf. upper rows for *bforig* with $k = 2$ in Table 5). Finally, we note that the size of inputs is primarily governed by conditions on PEOs and not significantly increased by adding labels.

To estimate the trade-off between filtering schemes and solution quality, Table 7 provides a summary of optimal network scores for the unlabeled and labeled *heart* and *econ*

Table 7 Quality of optimal Markov networks relative to different filtering schemes

Dataset	k	*bforig*	*node*	*part*	*edge*
heart	0, 1, 2	**−6,714.637**	**−6,714.637**	**−6,714.637**	**−6,714.637**
heart-lab	0, 1, 2	**−6,716.879**	**−6,716.879**	**−6,716.879**	**−6,716.879**
econ	0	−2,685.724	**−2,682.597**	**−2,682.597**	−2,688.747
	1	**−2,682.597**	**−2,682.597**	**−2,682.597**	−2,685.724
	2	**−2,682.597**	**−2,682.597**	**−2,682.597**	**−2,682.597**
econ-lab	0	−2,691.971	−2,690.404	−2,690.404	−2,695.188
	1	**−2,689.989**	−2,690.404	−2,690.404	−2,691.971
	2	**−2,689.989**	**−2,689.989**	**−2,689.989**	−2,690.404

datasets, where global optima (Nyman et al. 2014) are highlighted in boldface. For both *heart* variants, it turns out that optimal models are preserved regardless which scheme and parameter value are used for pruning clique candidates. This does not apply to the more complex *econ* dataset. In the unlabeled case, too small k-values decrease the solution quality with the *bforig* scheme, which does not take the contexts given by cliques into account, as well as the aggressive *edge* scheme. Given that the investigated filtering schemes assume empty labelings and do not consider potential alternatives, the usage of small k-values is particularly risky for the labeled *econ* variant. In fact, although the schemes *node* and *part* with $k \leq 1$ as well as the *edge* scheme with $k = 2$ come close to the global optimum, they still deteriorate the log marginal likelihood by 0.015 %. This suggests that labels should be taken into account in order to perform more aggressive yet informed pruning in the case of labeled Markov networks.

We further assessed the scalability of our approach on samples of the *hs* dataset of increasing size, using the *edge* scheme along with $k = 0, 1, 2$ for filtering clique candidates. The samples in the upper five rows of Table 8, with numbers of nodes given at the top, have been obtained by *max*imizing the density of the subgraph induced by a selection of the 25 nodes available in total, while the density is *min*imized in the lower five rows, with numbers of nodes given at the bottom. For both kinds of samples, the reported runtimes include the greatest numbers of nodes for which **CL*ASP***, again performing best when run as an ASP solver, was able to find and verify an optimal model within the time of one day. Similar to the *econ* dataset, the depth of filtering regulated by the parameter k has a significant impact on the resulting difficulty of Markov network structure learning. Moreover, samples of size 8 turn out to be much harder to solve than inputs obtained by applying the *edge* scheme with same k-value to the *econ* dataset, where the number of nodes is also 8. As the difficulty rapidly increases with sample size, optimization succeeds on dense subgraphs with up the 13, 10, or 9 nodes, respectively, depending on the k-value used for pruning. Although this value remains relevant, samples such that the density is minimized exhibit a considerably smoother scaling behavior and can be successfully handled up to 20, 18, or 15 nodes, respectively. This observation emphasizes the impact of problem structure as well as the importance of

Table 8 Solver runtimes in seconds on samples of the *hs* dataset with varying numbers of nodes and parameter values for the *edge* filtering scheme

	$k = 0$								$k = 1$					$k = 2$			
	6	7	8	9	10	11	12	13	6	7	8	9	10	6	7	8	9
max																	
CLASP	**0**	**2**	**55**	**204**	**1909**	**4856**	**15,420**	**65,670**	**1**	**17**	**144**	**1684**	**13,199**	**1**	**21**	**383**	**10,042**
CLASP	49	1604	38,441	85,279	*TO*	*TO*	*TO*	*TO*	1043	13,774	*TO*	*TO*	*TO*	476	26,493	*TO*	*TO*
PWBO	421	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	34, 298	*TO*	*TO*	*TO*	*TO*	16,765	*TO*	*TO*	*TO*
SAT4J	102	5293	*TO*	84,097	*TO*	*TO*	*TO*	*TO*	2387	*TO*	*TO*	*TO*	*TO*	2625	*TO*	*TO*	*TO*
CPLEX	38	642	21,400	*TO*	*TO*	*TO*	*TO*	*TO*	264	6662	*TO*	*TO*	*TO*	109	6049	*TO*	*TO*
min																	
CLASP	**0**	**3**	**11**	**42**	**151**	**485**	**5879**	**42,044**	**1**	**19**	**69**	**382**	**31,025**	**7**	**27**	**209**	**19,525**
CLASP	8	137	981	2218	6855	67,245	*TO*	*TO*	78	812	6789	27,334	*TO*	273	2370	17,253	*TO*
PWBO	66	1763	32,719	71,380	*TO*	*TO*	*TO*	*TO*	1478	17,123	*TO*	*TO*	*TO*	4712	*TO*	*TO*	*TO*
SAT4J	55	340	2144	13,550	60,848	*TO*	*TO*	*TO*	173	1841	17,139	*TO*	*TO*	714	6025	*TO*	*TO*
CPLEX	31	234	1781	3427	7147	*TO*	*TO*	*TO*	42	1136	5481	48,095	*TO*	689	9673	70,689	*TO*
	13	14	15	16	17	18	19	20	11	13	14	15	18	11	12	13	15

filtering in view of an input size (number of clique candidates) that, in the worst case, grows exponentially with the number of nodes.

7 Conclusion

Our experiments and the recent interest in graphical model learning using integer programming, maximum satisfiability, and answer set programming demonstrate that computational logic holds a largely untapped valuable resource for statistical inference. The main challenge lies in finding effective translations from the statistical learning problem into logical constraints, to make optimization scalable to large problem instances. As the diversity of approaches adopted here and in Bartlett and Cussens (2013), Berg et al. (2014), Corander et al. (2013), Cussens (2008), and Parviainen et al. (2014) shows, there is a lot of room for creativity in this translation task, and the choices made can strongly impact the performance of solvers. In addition, adopting strong pruning methods can be critical for reducing the space of candidate solutions. For instance, the approach based on dynamic programming (Kangas et al. 2014) suffers from exponential growth of memory consumption when the number of variables is increased.

We have first theoretically studied which models can be automatically recognized as inferior to others, such that the model space can be reduced without eliminating globally optimal models. Then, we introduced statistical pruning criteria for more extensive filtering of candidates, guaranteeing that best solutions are preserved asymptotically when the number of input data vectors increases. In future research it will be useful to study the effect of pruning strategies further and to develop translations of learning problems beyond graphical models.

Acknowledgements This work was supported by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170).

References

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

Berg, J., Järvisalo, M., Malone, B.: Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, pp. 86–95. JMLR.org (2014)

Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: Proceedings of the 12th International Conference on Uncertainty in Artificial Intelligence, pp. 115–123. Morgan Kaufmann (1996)

Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011)

Chandran, L.S., Ibarra, L., Ruskey, F., Sawada, J.: Generating and characterizing the perfect elimination orderings of a chordal graph. Theor. Comput. Sci. **307**(2), 303–317 (2003)

Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, pp. 151–158. ACM Press (1971)

Corander, J.: Labelled graphical models. Scand. J. Stat. **30**(3), 493–508 (2003)

Corander, J., Ekdahl, M., Koski, T.: Parallel interacting MCMC for learning of topologies of graphical models. Data Min. Knowl. Discov. **17**(3), 431–456 (2008)

Corander, J., Janhunen, T., Rintanen, J., Nyman, H., Pensar, J.: Learning chordal Markov networks by constraint satisfaction. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems, pp. 1349–1357. NIPS Foundation (2013)

Cussens, J.: Bayesian network learning by compiling to weighted MAX-SAT. In: Proceedings of the 24th International Conference on Uncertainty in Artificial Intelligence, pp. 105–112. AUAI Press (2008)

Dawid, A.P., Lauritzen, S.L.: Hyper-Markov laws in the statistical analysis of decomposable graphical models. Ann. Stat. **21**(3), 1272–1317 (1993)

Dellaportas, P., Forster, J.J.: Markov chain Monte Carlo model determination for hierarchical and graphical log-linear models. Biometrika **86**(3), 615–633 (1999)

Dirac, G.A.: On rigid circuit graphs. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg **25**(1–2), 71–76 (1961)

Eriksen, P.S.: Context specific interaction models. Technical Report, Department of Mathematical Sciences, Aalborg University (1999)

Eriksen, P.S.: Decomposable log-linear models. Technical Report, Department of Mathematical Sciences, Aalborg University (2005)

Friedman, N., Goldszmidt, M.: Learning Bayesian networks with local structure. In: Proceedings of the 12th International Conference on Uncertainty in Artificial Intelligence, pp. 252–262. Morgan Kaufmann (1996)

Galinier, P., Habib, M., Paul, C.: Chordal graphs and their clique graphs. In: Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science, pp. 358–371. Springer (1995)

Gebser, M., Janhunen, T., Rintanen, J.: Answer set programming as SAT modulo acyclicity. In: Proceedings of the 21st European Conference on Artificial Intelligence, pp. 351–356. IOS Press (2014)

Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence **187–188**, 52–89 (2012)

Giudici, P., Castello, R.: Improving Markov chain Monte Carlo model search for data mining. Mach. Learn. **50**(1–2), 127–158 (2003)

Giudici, P., Green, P.J.: Decomposable graphical Gaussian model determination. Biometrika **86**(4), 785–801 (1999)

Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York (1980)

Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bull. Am. Math. Soc. **64**(5), 275–278 (1958)

Højsgaard, S.: Split models for contingency tables. Comput. Stat. Data Anal. **42**(4), 621–645 (2003)

Højsgaard, S.: Statistical inference in context specific interaction models for contingency tables. Scand. J. Stat. **31**(1), 143–158 (2004)

IBM Corporation: IBM ILOG CPLEX Optimization Studio CP Optimizer User's Manual, version 12 release 6.0 edn. (2013)

Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci. **9**(3), 256–278 (1974)

Kangas, K., Koivisto, M., Niinimäki, T.: Learning chordal Markov networks by dynamic programming. In: Proceedings of the 28th

Annual Conference on Neural Information Processing Systems, pp. 2357–2365. NIPS Foundation (2014)
Kass, R., Raftery, A.: Bayes factors. J. Am. Stat. Assoc. **90**(430), 773–795 (1995)
Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. The MIT Press, Cambridge (2009)
Lauritzen, S.L.: Graphical Models. Oxford University Press, Oxford (1996)
Le Berre, D., Parrain, A.: The Sat4j library, release 2.2 system description. J. Satisf. Boolean Model. Comput. **7**, 59–64 (2010)
Lifschitz, V.: Answer set programming and plan generation. Artif. Intell. **138**(1–2), 39–54 (2002)
Madigan, D., Raftery, A.E.: Model selection and accounting for model uncertainty in graphical models using Occam's window. J. Am. Stat. Assoc. **89**(428), 1535–1546 (1994)
Manquinho, V., Marques-Silva, J., Planes, J.: Algorithms for weighted Boolean optimization. In: Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, pp. 495–508. Springer (2009)
Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm: A 25-Year Perspective, pp. 375–398. Springer (1999)
Martins, R., Manquinho, V., Lynce, I.: Parallel search for maximum satisfiability. AI Commun. **25**(2), 75–95 (2012)
Niemelä, I.: Logic programming with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. **25**(3–4), 241–273 (1999)
Nyman, H., Pensar, J., Koski, T., Corander, J.: Stratified graphical models-context-specific independence in graphical models. Bayesian Anal. **9**(4), 883–908 (2014)
Parviainen, P., Farahani, H.S., Lagergren, J.: Learning bounded tree-width Bayesian networks using integer linear programming. In: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, pp. 751–759. JMLR.org (2014)
Pensar, J., Nyman, H., Koski, T., Corander, J.: Labeled directed acyclic graphs: a generalization of context-specific independence in directed graphical models. Data Min. Knowl. Discov. **29**(2), 503–533 (2015)
Rose, D.J.: Triangulated graphs and the elimination process. J. Math. Anal. Appl. **32**(3), 597–609 (1970)
Sebastiani, R., Tomasi, S.: Optimization in SMT with LA(Q) cost functions. In: Gramlich, B., Miller, D., Sattler, U. (eds.) Automated Reasoning, pp. 484–498. Springer, Heidelberg (2012)
Silander, T., Kontkanen, P., Myllymäki, P.: On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In: Proceedings of the The 23rd Conference on Uncertainty in Artificial Intelligence (UAI-2007), pp. 360–367. AUAI Press (2007)
Silander, T., Roos, T., Kontkanen, P., Myllymäki, P.: Factorized NML criterion for learning Bayesian network structures. In: Proceedings 4th European Workshop on Probabilistic Graphical Models (PGM-2008) (2008)
Silander, T., Roos, T., Myllymäki, P.: Learning locally minimax optimal Bayesian networks. Int. J. Approx. Reason. **51**(5), 544–557 (2010)
Sinz, C.: Towards an optimal CNF encoding of Boolean cardinality constraints. In: Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, pp. 827–831. Springer (2005)
Whittaker, J.: Graphical Models in Applied Multivariate Statistics. Wiley, New York (1990)