Adding Local Constraints to Bayesian Networks

Mark Crowley[1], Brent Boerlage[2], and David Poole[3]

[1] University of British Columbia, Vancouver, BC, Canada. crowley@cs.ubc.ca
[2] Netica Division, Norsys Software Corp., Vancouver, Canada. boerlage@norsys.com
[3] University of British Columbia, Vancouver, BC, Canada. poole@cs.ubc.ca

Abstract. When using Bayesian networks, practitioners often express constraints among variables by conditioning a common child node to induce the desired distribution. For example, an 'or' constraint can be easily expressed by a node modelling a logical 'or' of its parents' values being conditioned to true. This has the desired effect that at least one parent must be true. However, conditioning also alters the distributions of further ancestors in the network. In this paper we argue that these *side effects* are undesirable when constraints are added during model design. We describe a method called *shielding* to remove these side effects while remaining within the directed language of Bayesian networks. This method is then compared to chain graphs which allow undirected and directed edges and which model equivalent distributions. Thus, in addition to solving this common modelling problem, shielded Bayesian networks provide a novel method for implementing chain graphs with existing Bayesian network tools.

Key Words: Bayesian networks, constraints, mixed networks, chain graphs, graphical models, Bayesian modelling, complementary priors

1 INTRODUCTION

When using Bayesian networks it is often convenient to use conditioned nodes to enforce constraints across the network. Consider the following example:

Example 1. There are three professors, **A**lice, **B**ob and **C**indy, at a university that needs at least one instructor for its AI course. For Alice we define four variables: I_A, modelling our belief that she is interested in AI; this influences W_A, our belief they she wants to teach the course; which influences T_A, our belief that she will actually end up teaching the course; which influences R_A, our belief that she completes her current research project on time. Variables are defined analogously for Bob and Cindy. The joint distribution of the variables T_A, T_B, T_C is consistent with the constraint that at least one professor must teach the course.

A natural way to represent this distribution is to add a node, C, to the network that models an 'or' of its parents and is conditioned to true. Figure 1 shows the Bayesian network that results. This enforces[4] the desired constraint onto the variables T_A, T_B, T_C

[4] There are other ways to achieve this type of distribution without conditioning but it requires many new variables to be added and is difficult to maintain, see [1] for more details.

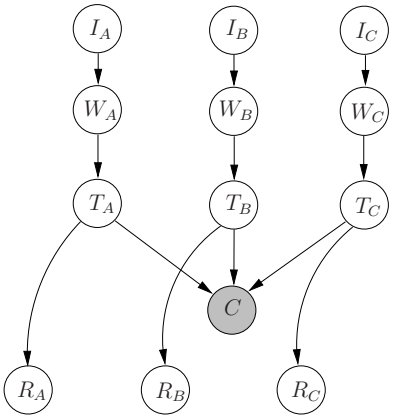

Fig. 1. Topic interests and teaching desires of three professors. C is an 'or' node stating that someone must teach the course.

which we call the *affected nodes*. This is similar to what [2] calls adding constraints using an auxiliary network. That paper models constraints by merging constraint network formalisms into Bayesian networks.

The desired distribution is one where the CPDs of all nodes express the probabilities *given* the presence of the constraint on the affected nodes. Thus, if $p(I_A = true) = .7$ then we want $p(I_A = true|C) = .7$. But in a standard BN this will not be the case. I_A will be influenced by C, we call this influence a *side effect* of C. The reason we don't want side effects is that they arise from treating C as evidence and $p(T_A|W_A)$ as a simple conditional distribution. In fact, for this model, neither of these is the case. C is merely a convenient way to express a constraint, it does not constitute evidence, and thus its value should not be used freely for inference amongst its ancestors. But the constraint must be satisfied amongst its parents and the distribution on T_A is defined given the constraint. $p(T_A|W_A)$ actually defines the probability distribution of T_A given that someone else has already been assigned to teach the course.

Thus, the constraint should have no influence no I_A. Our beliefs about Alice's interest in AI are tied to the likely teaching assignments but are decoupled from the constraints on those teaching assignments. Altering or observing Cindy's interest in AI should have no impact on Alice's interest. On the other hand, R_A, *is* influenced by the constraint. Research productivity is directly influenced by teaching assignments and so anything that impacts this must be taken into account when determining the likelihood of R_A.

Here we present a method to eliminate these side effects while maintaining a fully directed model and using existing Bayesian network tools. We will compare this method to chain graphs [3] which represent the same set of distributions by defining away the possibility of side effects. This paper has the dual goal of explaining how to solve a practical modelling problem with existing tools, as well as giving a new interpretation of chain graphs in terms of fully directed models.

2 BAYESIAN NETWORKS

A Bayesian network (BN) [4] is a directed acyclic graph that represents the interdependence amongst a set of random variables. Suppose the variables are $V_1, \ldots, V_n$. The Bayesian network represents the following the factorization for the joint probability of a set of nodes in a Bayesian network:

$$p(V_1, \ldots, V_n) = \prod_{i=1}^{n} p(V_i | pa(V_i)) \tag{1}$$

where $pa(V_i)$ are the parent nodes on which V_i is dependant, if any.

2.1 TYPES OF CONDITIONING

Conditioning refers to the general technique of setting a variable to a particular value within a BN. There are, at least, three types of conditioning. The most common type is simply recording an observation about the state of a variable or *observation conditioning*. The value here represents new information that rules out possible worlds that are incompatible with the observation. The remaining worlds are then renormalized to sum to 1. An observation can influence all of its ancestors and their descendants.. If a variable is set by the user arbitrarily we call this *intervention conditioning* [5]. In this case the variable is set to some value by a mechanism outside of the model and so is not indicative of the variable's distribution. Thus the intervention cannot be used for inference about influences on the variables Decision variables are of this type. An intervention should be cut off from influencing its ancestors but still influences its descendants.

A third type of conditioning, *constraint conditioning*, is the type being addressed in this paper. A node's value is set as part of the model definition in order to induce a particular distribution amongst its parent nodes. Other ancestors should not be affected just as they are not affected by the initial distributions of any other descendants. Influence on ancestors is cut off, just as in intervention, but in this case one level of nodes are allowed to be influenced. All of the descendants of these parents will then be influenced in the usual way. In this paper the constraint conditioned nodes such as C will be called *c-nodes*. The nodes in the constraint will (the parents) are the affected nodes or *e-nodes*. Nodes that are parents of affected nodes but are not themselves affected are known as *shielded nodes* or *s-nodes*

We believe this is an important modelling problem for BNs. Bayesian networks are used widely every day for a broad range of purposes. We know from discussions with practitioners and experience that constraint conditioning is often used in practice. This is done as a natural extension of BN modelling and the full ramifications of side effects on the model may not always be realized. It is important for this issue to be widely discussed and possible solutions or alternative modelling techniques provided.

3 REMOVING SIDE EFFECTS

Our goal now is to construct a BN in such a way that after inference is carried out the constraint conditioned nodes will have the desired influence and no more. We call this

method *shielding*. The chief insight is that we can add more conditioned nodes to cancel out the side effects. So, after adding a node $\hat{C}$, which we will define momentarily, we want the following to be true:

$$p(W_A|I_A, c, \hat{c}) = p(W_A|I_A) \qquad p(W_B|I_B, c, \hat{c}) = p(W_B|I_B)$$
$$p(W_C|I_C, c, \hat{c}) = p(W_C|I_C) \qquad p(W_A, W_B, W_C|c, \hat{c}) = p(W_A, W_B, W_C) \tag{2}$$

where c indicates that $C = c$.

3.1 ANTIFACTORS

To define $\hat{C}$ we need to think about how inference is carried out. A *factor* is the result of summing out some variables in a network during inference using a technique such as variable elimination [6][7]. In our example, if the affected nodes are summed out a factor is obtained, $f_{T_{ABC}}(W_A, W_B, W_C)$, representing the combined effect of the constraint on the W_A, W_B and W_C nodes. To cancel this we create an *antifactor* node, $\hat{C}$, with these nodes as parents, see Figure 2. The distribution of $\hat{C}$ is defined by inverting the factor for the affected nodes as follows:

$$p(\hat{c}|W_A, W_B, W_C) = \frac{1}{Z}\frac{1}{f_{T_{ABC}}(W_A, W_B, W_C)} \tag{3}$$

where $f_{T_{ABC}}(W_A, W_B, W_C) = \sum_{\mathbf{T}} p(c|T_A, T_B, T_C)p(T_A|W_A)p(T_B|W_B)p(T_C|W_C)$

The constant, Z, ensures that all values are in the range [0,1]. During inference this will cause the distributions of $\hat{C}$ and the nodes C, T_A, T_B and T_C to exactly cancel each other making the distribution consistent with (2).

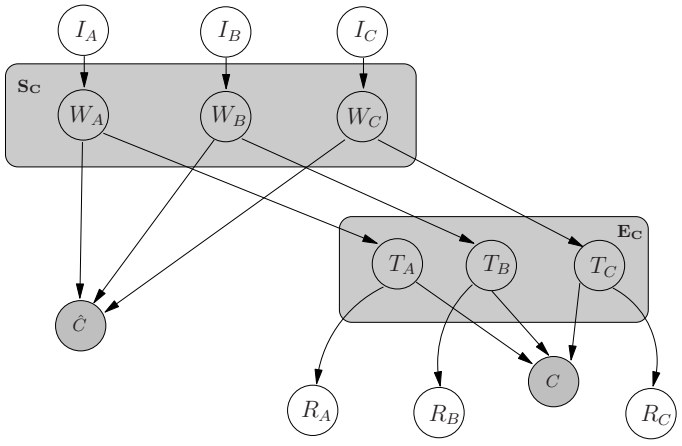

Fig. 2. An antifactor $\hat{C}$ shields the influence of the *c-node* C.

3.2 GENERAL ANTIFACTORS

We now define the problem more generally.

Definition 1. *A* shielded Bayesian network (SBN) *as satisfying the following requirement:*

$$p(\mathbf{S_C}|c, \hat{c}) = p(\mathbf{S_C}) \tag{4}$$

Where C is a c-node *and $\hat{C}$ is a conditioned node added to the network with a distribution constructed to satisfy* (4)*. The set $\mathbf{E_C} = pa(\mathbf{E_C})$ contains the* affected nodes *and $\mathbf{S_C} = pa(pa(C)) - \mathbf{E_C}$ contains the* shielded nodes*. We assume there is no node in that is both an ancestor and a descendant of nodes in $\mathbf{E_C}$.*

This can be satisfied by creating an antifactor node, $\hat{C}$, with parents $\mathbf{S_C}$ such that

$$p(\hat{c}|\mathbf{S_C}) = \frac{1}{Z}\frac{1}{f_{\mathbf{E_C}}(\mathbf{S_C})}$$

A more general case is shown in Figure 3. Here C_1 and C_2 are connected in a component because they share parents. Let κ be the minimum set of pairwise, disjoint components. The set $\mathbf{S}_\kappa$ then denotes all the nodes to be shielded from every *c-node* in κ. An antifactor, $\hat{C}$, is defined with parents $\mathbf{S}_\kappa$. Its distribution is computed by summing out all nodes in $\mathbf{E}_\kappa = E_{C_1} \cup E_{C_2}$.

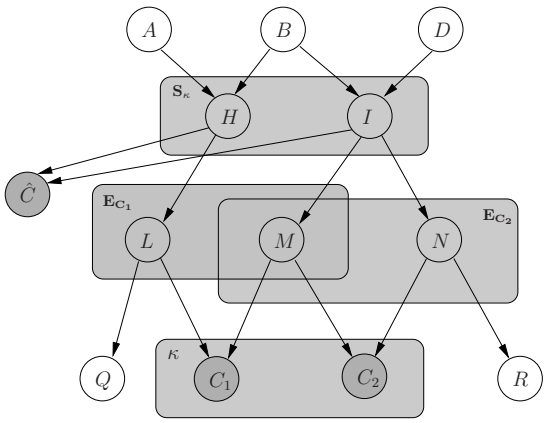

Fig. 3. The nodes L, M, N are constrained by two *c-nodes* with $\kappa = \{C_1, C_2\}$.The antifactor, $\hat{C}$, cancels out the effect on the $\mathbf{S}_\kappa$ nodes.

Definition 2. *With κ as a, possibly singleton, set of connected* c-nodes *and $\hat{C}$ as its corresponding antifactor node, the following is the general definition of shielding:*

$$p(\mathbf{S}_\kappa|\kappa, \hat{c}) = p(\mathbf{S}_\kappa) \tag{5}$$

Note that the antifactor *always exists* except in the case where the factor $f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)$ contains a zero term. This occurs when the distribution of the affected network assigns a probability of zero to some instance of $\mathbf{S}_\kappa$ after all the affected nodes have been summed out.

4 ANTINETWORKS

The major drawback of using antifactors is complexity. $\hat{C}$ connects all of the nodes in $\mathbf{S}_\kappa$ creating a large new clique in the network. We could improve complexity by creating a conditional structure to reduce the number of parents of the antifactor.

An *antinetwork* is a set of nodes that mimic the structure of the original *c-node* and its parents. The distributions of the copied $\hat{C}$ and $\mathbf{E}_{\hat{\kappa}}$ nodes are computed so that summing out $\mathbf{E}_{\hat{\kappa}}$ will yield $\frac{1}{f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)}$. Figure 4 shows the antinetwork for our example.

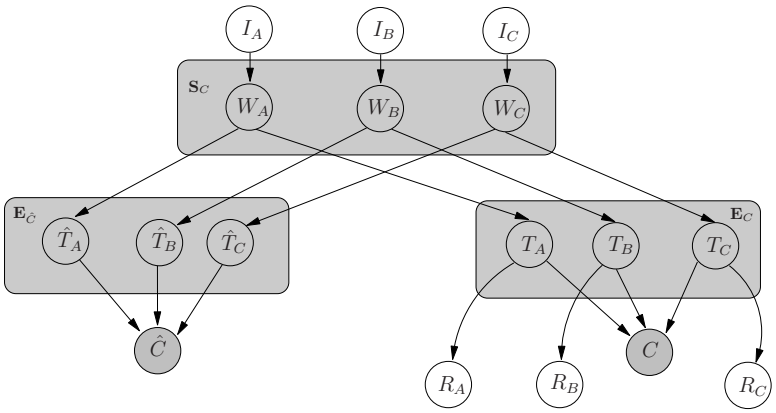

Fig. 4. An antinetwork shields the influence of *c-node* C.

4.1 EXISTENCE OF A SOLUTION

Unlike the antifactor solution it is not certain that a proper set of parameters for the antinetwork always exists although we have found them in many cases. Here we discuss some general properties of solutions.

The parameters to be solved for the antinetwork conform to the following system of equations. For simplicity, the case with binary nodes is shown here.

$$\begin{aligned} \textit{Let } \pi = f_{\mathbf{E}_{\hat{\kappa}}}(\mathbf{S}_\kappa) &= \frac{1}{f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)} \\ \pi &= \sum_{x \in \mathbf{E}_{\hat{\kappa}}} \prod_{\hat{c} \in \hat{\kappa}} p(\hat{c}|\mathbf{E}_{\hat{c}}) \prod_{\hat{e} \in x} p(\hat{e}|\mathbf{S}_{\hat{c}}) \\ 0 &= \sum_{x \in \mathbf{E}_{\hat{\kappa}}} \prod_{\hat{c} \in \hat{\kappa}} \gamma_{\hat{c}} \prod_{\hat{e} \in x} (\psi_{\hat{e}})^{(\hat{e}=\mathbf{t})} (1-\psi_{\hat{e}})^{(\hat{e}=\mathbf{f})} - \pi = g_s(X) \end{aligned} \tag{6}$$

Where x captures one assignment to all the nodes in $\mathbf{E}_{\hat{\kappa}}$ and the indicator exponent $(\hat{e} = \mathbf{t})$ is simply 1 or 0. Note that this represents one equation for each instance $s \in S_\kappa$. We will refer to this system as $g_s(X)$ for $X = \{\gamma_{\hat{c}}, \psi_{\hat{e}}\}$ for all $\hat{c} \in \kappa$ and $\hat{e} \in \mathbf{E}_{\hat{c}}$. When X is found such that $g_s(X) = 0$ then the antinetwork satisfies the shielding requirement.

Solution Bounds When all the parameters are set to zero, denoted X_0, and one, X_1, the system yields:

$$g_s(X_0) = -\pi \qquad g_s(X_1) = 1 - \pi$$

Since π is normalized to be a probability we have

$$g_s(X_0) \leq 0 \leq g_s(X_1) \qquad \text{for all } s \in \mathbf{S}_\kappa$$

Since $g_s(X)$ is a continuous function for each $s \in \mathbf{S}_\kappa$ we know there is a solution X_s such that $g_s(X_s) = 0$. Unfortunately, we have not yet been able to show that there is always a simultaneous solution, X_*, to these equations such that $g_s(X_*) = 0$ for all $s \in \mathbf{S}_\kappa$. The solution X_* is easy to identify when found as all the functions will be zero. When $X \neq X_*$, the solution can be used as an approximation to the correctly shielded distribution.

4.2 FINDING A SOLUTION

The antinetwork parameters can be solved by framing them as a *constrained optimization problem*. The objective function is a linear combination of the $g_s(X)$ functions. The same functions are also used to define nonlinear, inequality constraints of the form $g_s(X) \geq 0$. Optimization is then begun at some known positive point, such as X_1 and minimized until all $g_s(X) = 0$. See [1] for more details.

4.3 SOLUTION EXAMPLE

The solved CPDs for the antinetwork nodes, for Example 1, are shown in figure 5. The posteriors of the shielded nodes, W_A, and their ancestors, I_A, are correctly uninfluenced by the existence of the constraint. In particular, the ancestors maintain their prior distributions:

$$p(I_A = \mathbf{t}|c, \hat{c}) = .3 \qquad p(I_B = \mathbf{t}|c, \hat{c}) = .7 \qquad p(I_C = \mathbf{t}|c, \hat{c}) = .6$$

For networks where the affected sets overlap this method does not always find an exact solution. Our results approach the correct distribution but do not find an exact match. This indicates a solution may exist and that improved search techniques could yield a better approximation or an exact solution. When an exact solution is required an antifactor can always be used to shield the given *c-nodes* instead. Furthermore, antinetworks and antifactors can be used in the same network.

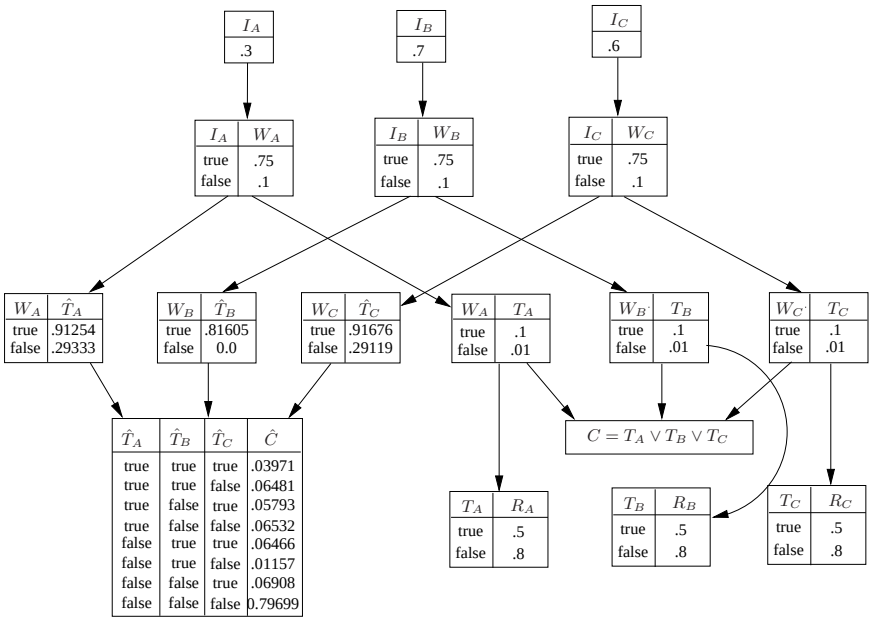

Fig. 5. Computed CPDs for antinetwork found using nonlinear constrained optimization.

5 UNDIRECTED MODELS AND CHAIN GRAPHS

Another way to think about constraint conditioning is through undirected models. A Markov Random Field (MRF) [4] can easily be expressed as a BN by replacing all cliques potentials, ϕ, with conditioned nodes. A simple construction makes this clear, see Figure 6:

- For each clique $\mathfrak{C}_\mathbf{i}$ in the MRF, remove all links between nodes and replace with a directed link from each node in $\mathfrak{C}_i$ to a new binary node C_i.
- Assign the CPD of C_i such that $p(C_i = \mathbf{true}|\mathfrak{C}_i) = \phi_i(\mathfrak{C}_i)$
- Condition all of these added C_i nodes to be true.

These two representations model equivalent distributions. Note that under this construction the conditioned nodes will never have grandparents so shielding will not be needed. To model the full range of distributions we are interested in we need a combination of directed and undirected relationships.

5.1 CHAIN GRAPHS

A *chain graph* (CG) [3] is a graphical model that can have directed or undirected edges between its nodes. A *chain component*, $\tau \in \mathcal{T}$, is any set of nodes forming a connected component using undirected edges. Nodes in the directed portions of the network form their own chain components of size one. A CG can be seen as a directed, acyclic graph

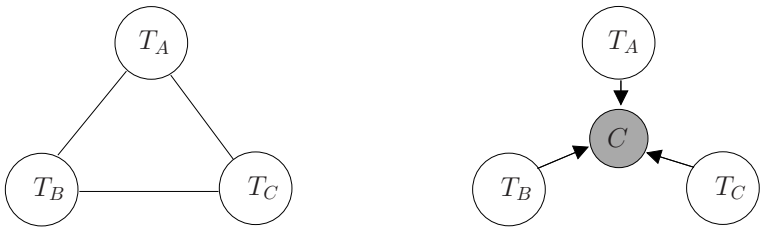

Fig. 6. MRF to BN construction: a) A Markov Random Field. b) This MRF as a Bayes net with conditioned nodes replacing clique potentials.

of chain components. The graph is acyclic in that there are no *partially directed cycles*. This is a cycle containing some directed edges, all pointing in the same direction around the cycle. Our example can be represented as a CG with the affected nodes represented as in figure 6(a) and other nodes connected as within the original BN.

The joint density of a CG is given by the following factorization [8] where the values of a set of variables $\mathcal{V}$ is given by $x_{\mathcal{V}}$. Here $\mathcal{A}(\tau)$ are all the fully connected sets of nodes from within $\tau \cup pa(\tau)$. Each of these has a clique potential $\phi_A(x_A)$ across the nodes in the fully connected set. The Z term normalizes the density by summing across the values of all the nodes within the current chain component:

$$p(x_{\mathcal{V}}) = \prod_{\tau \in \mathcal{T}} p(x_\tau | x_{pa(\tau)}) \tag{7a}$$

$$p(x_\tau | x_{pa(\tau)}) = \frac{1}{Z(x_\tau)} \prod_{A \in \mathcal{A}(\tau)} \phi_A(x_A) \tag{7b}$$

$$Z(x_\tau) = \sum_{x_\tau} \prod_{A \in \mathcal{A}(\tau)} \phi_A(x_A) \tag{7c}$$

Consider computing $p(W_A)$. It is clear that the distribution of the T_A, T_B, T_C chain component will play no part. This is because the nodes in the chain will be summed out in equation (7b) which will lead the ϕ and Z terms to cancel exactly. In fact, lacking any observations, the variables W_A, W_B, W_C are independent of each other. CGs thus already express the kind of distribution we are concerned with where a joint constraint can exist amongst a set of variables without their ancestors being affected by the existence of that constraint. Note that in the presence of observations of a node in T_A, T_B or T_C this independence would no longer hold as this is new information that is relevant to all nodes.

5.2 EQUIVALENCE OF SBNs AND CGs

We will show the equivalence of shielded Bayesian networks with chain graphs by mapping each portion of SBNs to the factorization of CGs from equation (7).

1. The assumption in definition 1 is equivalent to the restriction against partially directed cycles in CGs.
2. Each chain component, τ, in a CG has a corresponding set κ of *c-nodes* in an SBN.
3. Each potential function in the CG corresponds to the distribution of a *c-node* and the set of its affected nodes in the SBN

$$\phi_A(x_A) \propto p(c|\mathbf{E_C})p(\mathbf{E_C}|\mathbf{S_C}).$$

4. The Z term is equivalent to marginalizing out the affected nodes. So for antifactors:

$$\frac{1}{Z(x_\tau)} \propto p(\hat{c}|\mathbf{S}_\kappa) = \frac{1}{f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)}. \tag{8}$$

   And similarly for antinetworks:

$$\frac{1}{Z(x_\tau)} \propto p(\hat{c}|\mathbf{E}_\kappa) \prod_{\mathbf{E}_\kappa} p(\mathbf{E}_\kappa|\mathbf{S}_\kappa) = \frac{1}{f_{\mathbf{E}_\kappa}(\mathbf{S}_\kappa)} \tag{9}$$

   Note that if κ contains more than one *c-node* then they must be dealt with simultaneously.

With these mappings in place, the joint distributions in either model comes from a calculation that is equivalent up to a constant factor. This equivalence shows us that both models can be used to represent the same distribution.

The complexity of inference in graphical models is exponential in the size of the largest cliques in the network. We use clique in the same sense as in Junction trees [9], which are often used to perform inference in graphical models. Using either SBNs with antifactors or CGs this will be dominated by the size of the set $\mathbf{S}_\kappa$. This is because an antifactor has all of the nodes in $\mathbf{S}_\kappa$ as its parents and so creates a clique of that size. Likewise, the potential function of a CG, $\phi_A(x_A)$, is defined over a *moralized graph* [10] where all the parents of nodes in a chain component are connected together. As we will show in the next section, antinetwork can avoid this blowup at least for some classes of networks.

6 COMPLEXITY COMPARISON

Consider the case where each *e-node* has m parents, none of which are shared with other *e-nodes* and all nodes have a domain of size D. This is the type of distribution described in Example 1. In all networks of this type tried an antinetwork solution has always been found.

The complexity for CGs is then exponential in the size of the clique A which is :

$$\begin{aligned} CG &= D^{|E_C|+|S_C|} \\ &= D^{|E_C|+m|E_C|} \qquad \text{since } |S_C| = m \times |E_C| \\ &= D^{|E_C|(m+1)} \end{aligned} \tag{10}$$

For SBNs with antifactors, all of the *s-nodes* are combined into one clique. To maintain the triangulation property for junction trees each *e-node* is joined to all *s-nodes*. This leads to a slightly higher complexity than for CGs although the dominant term is the same as the CG complexity.

$$SBN_{\text{antifactor}} = D^{|E_C|+|S_C|} + D^{|S_C|} + D^{|E_C|} \tag{11}$$

An example of the junction tree for the third model, using antinetworks, is shown in figure 7. Conditional independence in the antinetwork reduces the complexity to:

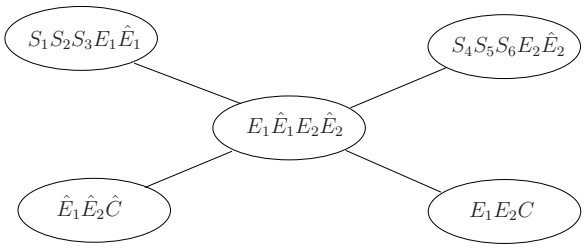

Fig. 7. Junction tree for antinetwork model with $|\mathbf{E_C}| = 2$ and $m = 3$.

$$SBN_{\text{antinetwork}} = D^{2|E_C|} + |E_C|D^{2+m} + 2D^{|E_C|} \tag{12}$$

Thus, for this set of models we find that

$$SBN_{\text{antinetwork}} < CG \quad \text{when both } |E_C| \geq 2 \text{ and } m \geq 2.$$

So in general, as the number of parents of each *e-node* goes up, SBNs increase in complexity more slowly than CGs if the connectivity between ancestors of each *e-node* is low. When this is not the case, the antifactor method still provides a solution that has the same dominant term as the CG although it will have additional, smaller cliques as well.

7 CONCLUSION

In this paper we have formalized a common informal technique for adding constraints to BNs and pointed out serious side effects that may not be desired. The modeler faced with these unwanted side effects has several choices. They could re-evaluate their modelling assumptions, attempt to represent the constraint in other ways or use chain graphs instead. Modelers now have another option which is to use one of the shielding methods proposed here. The first method, antifactors, is universal and simple to apply but may be costly during inference. The second method, antinetworks, is more efficient for inference and while the empirical existence of solutions is promising there are no guarantees as of yet. The distributions modelled by these networks are equivalent to those of

chain graphs. We have shown that at least for some classes of distribution, antinetworks are a more efficient representation than chain graphs. Further questions remain such as: Are there antinetwork solutions for wider classes of BNs? Are there any distributions that have compact antifactor solutions that would combine the advantage of both shielding methods? Can antifactors or antinetworks take advantage of context specific independence to reduce complexity?

There are strong similarities between our methods and complementary priors in [11] which offer intriguing lines of further research into learning. That work computes complimentary priors quickly and efficiently to cancel out interdependence between network layers, it would be interesting to see if this can be applied to our modelling task. We believe the modeller's toolkit should include a variety of methods that allow flexibility to model any distribution needed. The techniques described here can be a very useful part of that toolbox when directed models and constraints are needed. Chain graphs are also available for these tasks but SBNs would be very natural to many modellers familiar with BNs. They require no extra tools beyond standard BN software to be used and it would be straightforward to implement precompilers to automatically add antifactors or antinetworks to BNs. This solution contributes to Bayesian network modelling as well as adding insight into the relationships between all of these common modelling languages.

References

1. Crowley, M.: Shielding against conditioning side effects in graphical models. Master's thesis, University of British Columbia, Canada (October 2005)
2. Dechter, R., Mateescu, R.: Mixtures of deterministic-probabilistic networks and their and/or search space. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-04), Arlington, Virginia, AUAI Press (2004) 120–129
3. Lauritzen, S.L., Wermuth, N.: Graphical models for associations between variables, some of which are qualitative and some quantitative. Annals of Statistics **17** (1989) 31–57
4. Pearl, J.: Probabilistic Reasoning in Intelligent Systems:Networks of Plausible Inference. Morgan Kaufmann (1988)
5. Pearl, J.: Graphical models, causality and intervention. Statistical Science **8** (1993) 266–269
6. Zhang, N., Poole, D.: Exploiting causal independence in bayesian network inference. Journal of Artificial Intelligence Research **5** (1996) 301–328
7. Dechter, R.: Bucket elimination : A unifying framework for probabilistic inference. In Horvitz, E., Jensen, F., eds.: Proceeding of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96). (1996) 211–219
8. Lauritzen, S.L., Richardson, T.S.: Chain graph models and their causal interpretations. Journal of the Royal Statistical Society **64**(3) (2002) 321–361
9. Jensen, F.V.: Junction trees and decomposable hypergraphs. Technical report, Judex Datasystemer, Aalborg, Denmark. (1988)
10. Lauritzen, S.L.: Graphical Models. Oxford:Clarendon (1996)
11. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comp. **18**(7) (July 2006) 1527–1554