Shielding Against Conditioning Side Effects in Graphical Models

by

Mark Anthony Crowley

B.A., York University, 1999

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University of British Columbia

October 2005

Abstract

When modelling uncertain beliefs with graphical models we are often presented with "natural" distributions that are hard to specify. An example is a distribution of which instructor is teaching a course when we know that someone must teach it. Such distributions over a set of nodes can be easily described if we condition on a child of these nodes as part of the specification. This conditioning is not an observation of a variable in the real world but by fixing the value of the node, existing inference algorithms perform the calculations needed to achieve the desired distribution automatically. Unfortunately, although it achieves this goal it has side effects that we claim are undesirable. These side effects create dependencies between other variables in the model. This can lead to different beliefs throughout the model, including the constrained variables, than would otherwise be expected if the constraint is meant to be local in its effect. We describe the use of conditioning for these types of distributions and illuminate the problem of side effects, which have received little attention in the literature. We then present a method that still allows specification of these distributions easily using conditioning but counterbalancing side effects by adding other nodes to the network.

Table of Contents

List of Tables

List of Figures

Acknowledgements

Gratitude is something I'm full of right now, many people have contributed to this thesis though they may not all know it. To my parents, you have made me who I am. My love of knowledge, desire to do things the right way or not at all and dependancy on chocolate are all because of you and I couldn't have a better, more loving pair of parents. You always support me, even if my choices don't always seem to make too much sense at first. Thank you.

To my wife Lily, you know I wouldn't have been able to do this without you, and believe me I know it too. You motivate me and you calm just me just when I need it, I know I'm the luckiest man in the world to have you.

To the many people around LCI that have helped me with my ideas over the past year especially Kevin Murphy, Kevin Leyton-Brown, Mike Klaas, Michael Chiang, Rita Sharma and Jacek Kisynski. You may not realize how much your comments helped, but I often think to questions and observations each of you has said, and they've all been helpful.

To Brent Boerlage, your question got this all started so I definetly wouldn't be here without that, and your help writing the UAI paper was invaluable in the development of the ideas behind this paper. To my supervisor David Poole, your insightful ideas and piercing questions always stir new courses of inquiry and exciting solutions. I have been honoured to be able to work you and look forward to continuing to work on all those great ideas locked up in your mind.

And to the complexity and beauty that fills the universe, that is the source of all interesting ideas, including life. I know in some way I have been helped along so far on this swerving road of life by your truth, so, thank you too.

To Lily,
For Lily.

Chapter 1

Introduction

The essence of knowledge is, having it, to apply it;
not having it, to confess your ignorance.
$\sim$ *Confucius*

In directed graphical models conditioning of nodes to particular values usually corresponds to an observation of the value of some variable in the real world. There is another use of conditioning, however, that is often applied but seldom discussed in the literature. That is, to specify joint distributions or constraints across the parents of the conditioned node. This method is a straightforward way to specify constraints as many computations are handled automatically by the existing inference algorithms for graphical models. This thesis discusses the properties of conditioning including the possibly undesired creation of dependencies in the network not implied by the original constraints called *conditioning side effects*. We present a method to counteract side effects efficiently while maintaining the effectiveness and expressive simplicity of conditioning.

The nature of this topic is intimately tied to the language and rules of graphical models and Bayesian probability. As such, it is difficult to delve deeply into a definition until the appropriate background material has been covered, this will be done in chapter 2. For now, we begin with a simple, intuitive example to provide an idea of the topic and some general motivation for why it is important.

Example 1 *Consider a model of three university instructors, Alice, Bob and Cindy, and an Artificial Intelligence (AI) course. The model has the following components:*

1. *An instructor's interest in the topic of AI: The binary variable I_X represents whether or not instructor X is interested in the topic of AI.*

2. *Whether an instructor wants to teach the AI course: This is modelled by the binary variable W_X for instructor X. The belief that X wants to teach the course is dependant upon their interest in the topic, I_X.*

3. *A constraint on teaching : At least one instructor must teach the course.*

4. *Whether an instructor will teach the course: This is modelled by the binary variable T_X for instructor X. Our belief that X will teach the course is dependent upon whether the instructor actually wants to teach it, W_X, and whether someone else is already teaching the course.*

5. *Research productivity of instructors : This is modelled by the binary variable R_X for instructor X. If R_X is true it means that instructor X will complete their research goals for this semester. Our belief in the value of R_X is dependent upon whether or not they end up teaching the AI course, T_X, which will reduce the time they can spend on research.*

The frame of reference for this work is the knowledge engineering task of the person who must model a distribution such as the one above. The distributions we are in interested have the following two distinct components across some subset of the variables:

undirected component : This is a constraint on the values of the set of variables, or any type of undirected, joint distribution over them. In example 1 this type of distribution is specified in point 3 as a simple "or" of the states of the variables T_A,T_B and T_C. This is a simultaneous constraint. It is fundamentally not a directed relationship, the teaching status of each instructor affects all of the others.

directed component : Each constrained variable also has a directed component which specifies its distribution when the constraint is satisfied by some other variable. The directed component of the model of these variables is expressed in the example by points 4.

The goal is to combine these two components to produce a coherent model. We show in chapter 3 that there are ways to combine the two by forcing a directionality onto the undirected component. We argue that conditioning is a much more natural method that is very likely to be used by knowledge engineers. This method takes advantage of existing properties of graphical models to easily specify these distributions. It involves creating an artificial variable, fixed to one value to induce the constraint onto the variables. We show how this technique can be used in directed graphical models, such as Bayesian networks. We also highlight a problem with this technique which is not discussed in the literature.

If the constrained variables are conditionally dependent on other variables in the model then conditioning creates new dependencies between all of the influencing variables. These dependencies are not necessarily implied by the desired distribution and are called *side effects of conditioning*. In example 1 the use of conditioning to combine points 3 and 4 creates dependencies amongst the I_X and W_X variables. This could allow us to reduce our belief that Bob is interested in AI if we find out that Cindy is interested in AI. But the distribution for example 1 does not support this inference. Cindy and Bob's interests are unconditionally independent from each other. Only the teaching variables are constrained. It is perfectly possible that they are all interested in AI or that they all find it quite dull. The new dependencies tell us however that conditioning induces a belief that their is a kind of total level of interest in AI. If some instructors are less interested then it must be that others are more interested, since one of them must teach the course. This is not supported by the desired distribution for example 1. The probabilities for teaching are already conditioned on the fact that the constraint is satisfied by someone else.

The contribution of this thesis is to illuminate this subtle problem and provide a solution for it. We show that a conditioned node used in this way is fundamentally different from a normal node with evidence entered that corresponds to the observation of some variable. It turns out that in the types of distributions we've described it is often the case that the constraint is meant to remain local in its effect. In chapter 3 we will define the method of conditioning used to induce these constraints or undirected distributions. The notion of side effects of conditioning is explored fully in chapter 4. In chapter 5 we define how side effects can be counteracted through a process called *shielding*. A technique for implementing this shielding without unnecessary cost to the complexity of inference is presented in the form of *cloned anti-networks* in chapter 6. This is followed by a chapter containing implementation details and results on several example networks.

Chapter 2

Background

We now provide a brief overview of notation used in this work and probabilistic graphical models. If you are already familiar with graphical models you may want to only skim this chapter. For others it hopefully provides a good introduction to the area.

2.1 Variables, Nodes and Sets

We begin with the building blocks of all graphical models, random variables. These are represented as circular nodes in a graph. Each variable, X, in the graph has a domain of possible values it can take on, $dom(X)$

Random variables, or nodes in a network, are presented in capitalized italics whereas sets of nodes are in boldface, as in $C_i \in \mathbf{C}$. Specific values of nodes are shown in lowercase italics, such as c_i to mean $C = c_i$. When the domain of a variable is Boolean we simply use the lowercase name for true and false as in c_i and $lnotc_i$ to mean $C = true$ and $C = false$ respectively.

For a set of nodes $\mathbf{X} = \{X_1, \ldots, X_n\}$ we define the domain of the set to be

$$dom(\mathbf{X}) = X_1 \times X_2 \times \cdots \times X_n \tag{2.1}$$

For a set of nodes in a graphical model this means that each $\mathbf{x} \in dom(\mathbf{X})$ is a unique assignment of values to all the nodes in $\mathbf{X}$.

2.2 Bayesian Networks

A *Bayesian Network* (Bayes net, belief network) [16] is a directed, acyclic graph with a node for each random variable. The network defines a probability distribution over the set of random variables, $\mathbf{X} = \{X_1, \ldots, X_n\}$. The set of *parents* of a node X_i, denoted $pa(X_i)$, are all the variables on which X_i is conditionally dependent. This is indicated in the graph by directed arcs going from each element of $pa(X_i)$ into X_i, see figure 2.3. We refer to the ancestors of a node, $anc(X_i)$ as the transitive closure of $pa(X_i)$, thus any node reachable from X_i going upwards only in the graph. The descendants of X_i are defined similarly, $desc(X_i)$ is all the nodes reached by following links downwards in the graph.

Each node X_i has an associated *conditional probability distribution* (CPD) over X_i and its parents defining $p(X_i|pa(X_i))$. The structure of the graph encodes a statement of independence that a node is independent of all its non-descendants given the values of its parents. This CPD can be represented by a

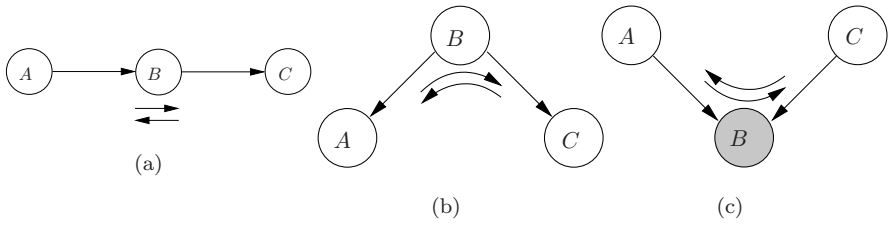


Figure 2.1: Structures that allow effects to pass through

table which has a number of entries that is exponential in the number of parents of the node. If a node has no parents then a prior distribution is specified as $p(X_i)$ and we define that $p(X_i|pa(X_i)) = p(X_i)$.

The joint probability distribution of the entire network is then given by the chain rule

$$P(\mathbf{X}) = \prod_{X_i \in \mathbf{X}} p(X_i|pa(X_i)). \tag{2.2}$$

2.2.1 Inference in Bayesian Networks

The task of inference is to compute the marginal probability of a node, $p(X_i|e)$, given the distribution modelled by the network and any evidence, e, which may have been observed. When the state of a variable, X_i, is observed this is entered as evidence in the Bayes net by fixing the value of X_i to the one observed and shading the node in the graph. The updated marginal probability of a node is then given in general by

$$p(X_i) = \sum_{\mathbf{X}-X_i} p(X_1, \ldots, X_n). \tag{2.3}$$

How the effect of evidence propagates throughout the Bayes net is very important for the discussions that follow. It can be described graphically with a simple set of rules called the Bayes Ball algorithm [19].

We can think of the probabilistic influence as propagating outwards in all directions along the edges connected to each node. Figures 2.1 and 2.2 show the three basic structures from which all Bayes nets can be built and the effect of evidence as it propagates through the network. Case 2.1(c), called a *v-structure*, is particularly important for the topic of this work. It shows that the parents of conditioned node become interdependent. This feature is what allows us to use conditioning to induce joint distributions. In conjunction with cases 2.1(a), 2.1(b) and 2.2(c) the computed beliefs of nodes throughout the network can be affected. If two nodes are blocked from affecting each other according to these rules then the nodes are said to be *d-separated* [16].

There are many algorithms used to carry out inference in Bayes nets, for good overviews see [13][4][22]. We describe two of them here that are pertinent

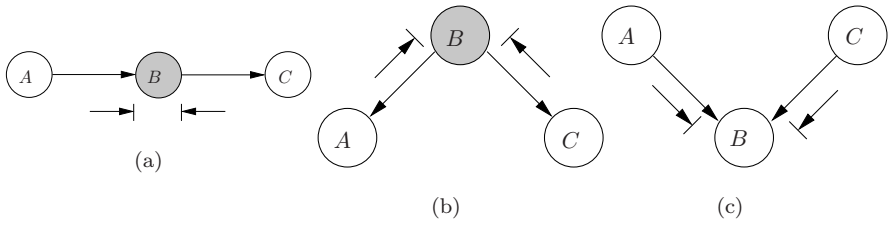


Figure 2.2: Structures that do not allow effects to pass through.

to our discussion.

2.2.2 Variable Elimination

Variable elimination [23][5] is a method of performing inference which is used in this work. The method essentially comes down to recognizing that the posterior distribution of a query node, G, can be computed by using the joint probability of the entire network and summing out all nodes other than the query node.

Given a query node G, a set of observations entered as evidence into nodes $\mathbf{e}$ and m remaining nodes $\mathbf{Y} = \mathbf{X} - \{G\} - \mathbf{e}$ we have:

$$p(G|\mathbf{e}) = \frac{p(G,\mathbf{e})}{p(\mathbf{e})} = \frac{p(G,\mathbf{e})}{\sum_G p(G,\mathbf{e})} \tag{2.4}$$

where (2.5)

$$p(G,\mathbf{e}) = \sum_{Y_1,\ldots,Y_m} p(G,\mathbf{e},Y_1,\ldots,Y_m) \tag{2.6}$$

In a given Bayes net, then, inference can proceed by summing out, or *eliminating*, all unobserved, non-query variables. The algorithm by Zhang & Poole [23] uses *factors* to store the intermediate sums in this process. The order of elimination that is chosen can have a significant impact on the complexity of the algorithm though it does not affect the correctness of the final answer.

Example 2 *In the network shown in figure 2.3, the elimination order used is* A E F C D. *The computation of the marginal probability of query node G given evidence b is shown in* (2.7).

The factors produced at each step are subscripted with the nodes that have been eliminated to produce them and have the remaining free variables in brackets. So for example, once the node A is eliminated it produces the factor $f_A(C)$. The factors are represented as tables just as the CPDs for nodes but factors have no conditional meaning. Factors can be multiplied together with CPDs

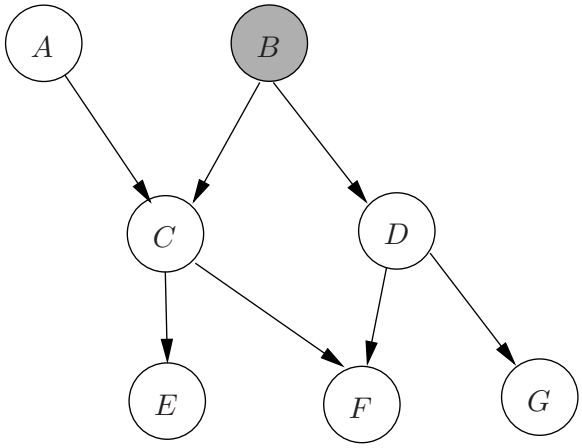


Figure 2.3: An example Bayesian network with node B conditioned to true.

or other factors to produce new factors in the same way that CPD tables are multiplied, element by element given the elimination ordering. The algorithm proceeds accoring to the following equation:

$$
\begin{aligned}
p(G,b) &= \sum_{A,C,D,E,F} p(A,b,C,D,E,F,G) \\
&= \sum_{A,C,D,E,F} p(G|D)p(D|b)p(F|C,D)p(E|C)p(C|A,b)p(A)p(b) \\
&= \sum_{D} p(G|D)p(D|b) \sum_{C} \sum_{F} p(F|C,D) \sum_{E} p(E|C) \sum_{A} p(C|A,b)p(A)p(b) \\
&= \sum_{D} p(G|D)p(D|b) \sum_{C} \sum_{F} p(F|C,D) \sum_{E} p(E|C) f_A(C) \\
&= \sum_{D} p(G|D) \sum_{C} \sum_{F} p(F|C,D) f_{AE}(C) \qquad (2.7) \\
&= \sum_{D} p(G|D) \sum_{C} f_{AEF}(C,D) \\
&= \sum_{D} p(G|D) f_{ACEF}(D) \\
&= f_{ACDEF}(G)
\end{aligned}
$$

We appeal to the intuition of this algorithm throughout our work and to the concept of factors. For efficient implementations we are interested in the following algorithm.

2.2.3 Junction Tree Algorithm

The junction tree algorithm [11][9] performs essentially the same calculations as the variable elimination approach but avoids duplicated computations over multiple queries through the use of a tree structure for caching. The tree is constructed based on the structure of the Bayes net by a method called *triangulation* which has two steps. The first step involves creating undirected arcs between all nodes with common children in the BN. This is known as *moralization* since all the parents in the network are being forced to "marry". In the second step, additional arcs are added to the graph until there are no remaining cycles with more than three nodes without arcs crossing between nodes in the cycle. The resulting graph, with the directionality of any arcs removed, is called fully triangulated or *chordal*.

This graph is then used to construct a tree by finding all of the maximal cliques in the graph and creating a node in the tree labelled with the nodes in that clique. A clique is a set of nodes such that each node in the set is connected to every other node in the set. A clique is maximal if it is not a subset of any other clique in the graph. The clique-nodes in the tree are furthermore connected in such a way so as to satisfy the *junction tree property*. This is simply that if any two clique-nodes in the tree contain some variable X, then X must also be present in all clique-nodes on the connecting path between them.

The algorithm passes messages between clique-nodes that represent the marginal distributions of the variables those cliques have in common. This is continued until all neighbouring cliques agree on their marginal distributions, generally in two passes through the tree. At this point all of the marginal distributions of the nodes in the network are cached in the clique-nodes of the junction tree, ready to be queried efficiently. This algorithm is widely used in many implementations of Bayesian networks because of its speed and efficiency in many practical cases. The complexity of the algorithm, however, is still exponential in the size of the largest clique in the network. Thus, minimizing the size of the largest clique is very important, an issue we return to for our solution in section 5.2.2.

2.3 Related Work

The work presented here clearly builds on the prodigious literature of graphical models as a whole and deals with a fundamental feature of them, conditioning on evidence. Very little in that literature, however, refers to the two fundamental concepts of this work, using conditioning to specify a distribution and side effects that arise from such conditioning.

There is also a strong relationship between the expression of joint distributions using conditioning and undirected distributions such as those in Markov Random Fields [16]. Work has been done by Buntine on chain graphs [3] attempting to merge directed and undirected models. The use of conditioning has similar possible applications for using both modelling paradigms. This work fo-

cusses on a fully directed representation whereas Buntine's always grounds back to undirected models. Our work also relies fully on existing inference algorithms for directed graphical models rather than devising a new one. Yedidia et al. [22] present the MRF-BN relationship using factor graphs which has similarities to the affect of conditioned nodes in a network. They do not discuss, however, the use of conditioning as a modelling tool as we do.

The literature on constraint satisfaction, see [12], is also somewhat related as *c-nodes* can be used as constraints. Their application goes beyond constraints however to the creation of any undirected joint distributions.

Chapter 3

Using Conditioning to Induce Distributions

We begin with a simplified version of example 1 without the I_X or W_X variables in order to focus on the use of conditioning without the concern of side effects. We describe three different ways to model the types of distributions discussed in the introduction using Bayesian networks. In these models we have a prior, conditional distribution for some nodes that applies when a constraint is satisfied or not present. We must distinguish these priors, which are provided to the modeler, from the distributions actually entered into the Bayesian network in order to achieve the correct, constrained distribution. We use $Pr(X)$ to denote the prior distribution of node X and $p(X)$ to refer to the CPD used within the Bayes net to model the combined distribution for X.

Example 3 *Consider a model of three university instructors, Alice, Bob and Cindy, and an Artificial Intelligence (AI) course. The model has the following components:*

1. *A constraint on teaching : At least one instructor must teach the course.*

2. *Whether an instructor will teach the course: This is modelled by the binary variable T_X for instructor X. Our belief that X will also be teaching the course once the constraint has been satisfied by another instructor is specified as a prior distribution, $Pr(T_X)$, the same for each instructor:*

$Pr(t_X)$	$Pr(\neg t_X)$
.1	.9

3. *Research productivity of instructors : This is modelled by the binary variable R_X for instructor X. If R_X is true it means that instructor X will complete their research goals for this semester. Our belief in the value of R_X is dependent upon whether or not they end up teaching the AI course, T_X, which will reduce the time they can spend on research. This belief is represented by the distribution $Pr(R_X|T_X)$, and is the same for each instructor:*

T_X	$Pr(r_X\|T_X)$	$Pr(\neg r_X\|T_X)$
true	.5	.5
false	.8	.2

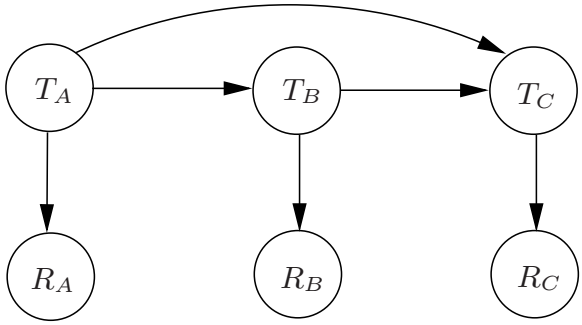


Figure 3.1: A Bayesian network with a directed clique to model example 3.

The constraint in this example has the effect of making the T_X nodes completely interdependent. We would expect that since at least one of the T_X variables is true that all other things being equal they should each have at least a $\frac{1}{3}$ chance of teaching the course plus there additional likelihood of being the second or third instructor on the course. If we observe that, for example, Alice is not teaching the course, then the responsibility for satisfying the constraint is spread between Bob and Cindy giving them each a probability of at least $\frac{1}{2}$. Our beliefs in the research productivity of each instructor is influenced by these changes as well. So our model specifies that Alice not teaching the course reduces our belief that Bob or Cindy will be productive since it is more likely they will be busy teaching.

3.1 Method 1 : Directed Clique

Since the nodes T_A, T_B, T_C are interdependent we can think of modelling them with a fully connected clique of nodes, allowing each node to have information about all the other constrained nodes. Directed models are acyclic so we need to force a direction onto the constraint by specifying an arbitrary ordering and computing the appropriate CPD for each constrained node based on the states of all "previous" nodes. For the order (T_A, T_B, T_C), the final node, T_C, has all the other constrained nodes as its parents, as shown in figure 3.1.

To compute the CPDs for this network it helps to consider that we are constraining the joint distribution of the three nodes so we can view the model as the Bayesian network shown in figure 3.2 with one node for the marginal probability of the teaching variables. The marginal of the prior distributions are shown in table 3.1(a)

Now we need to apply the constraint to this $< T_A, T_B, T_C >$ node. The "or" constraint in example 3 makes the case where all three variables are $false$ in table 3.1(a) infeasible, so we replace it with probability zero. We renormalize this to yield $p(T_A, T_B, T_C)$ shown in table 3.1(b).

We can now compute the CPDs to for the directed clique structure in the

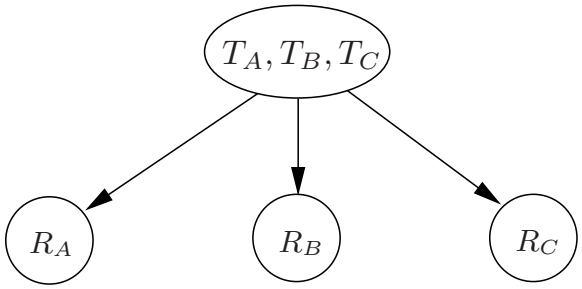


Figure 3.2: Bayes net for example 3 modelling the joint distribution of nodes T_A, T_B and T_C.

T_A	T_B	T_C	$Pr(T_A, T_B, T_C)$
t	t	t	$.1 \times .1 \times .1 = .001$
t	t	f	$.1 \times .1 \times .9 = .009$
t	f	t	$.1 \times .9 \times .1 = .009$
t	f	f	$.1 \times .9 \times .9 = .081$
f	t	t	$.9 \times .1 \times .1 = .009$
f	t	f	$.9 \times .1 \times .9 = .081$
f	f	t	$.9 \times .9 \times .1 = .081$
f	f	f	$\mathbf{.9 \times .9 \times .9 = .729}$

(a) Computed marginal of the prior distributions, $Pr(T_A, T_B, T_C)$. This is before the constraint is applied. The case (f, f, f) is inconsistent with the "or" constraint.

T_A	T_B	T_C	$p(T_A, T_B, T_C)$
t	t	t	.0037
t	t	f	.0332
t	f	t	.0332
t	f	f	.2989
f	t	t	.0332
f	t	f	.2989
f	f	t	.2989
f	f	f	0.0

(b) Posterior distribution $p(T_A, T_B, T_C)$ of the teaching nodes after the constraint is applied.

Table 3.1: Marginal probabilities for teaching variables prior and posterior to the application of the constraint

Bayes net shown in figure 3.1 by marginalizing out the appropriate variables as follows and then normalizing:

$$p(T_A) \propto \sum_{T_B, T_C} p(T_A, T_B, T_C) \tag{3.1a}$$

$$p(T_B|T_A) \propto \sum_{T_C} p(T_A, T_B, T_C) \tag{3.1b}$$

$$p(T_C|T_A, T_B) \propto p(T_A, T_B, T_C) \tag{3.1c}$$

Remember that this needs to be done just to specify a distribution consistent with example 3, we are actually performing inference on portions of the network during specification. The complete conditional probability tables are shown in figure 3.3. After performing inference the computed beliefs are as shown in figure 3.4 as displayed in the Bayesian network software package Netica [7]. All

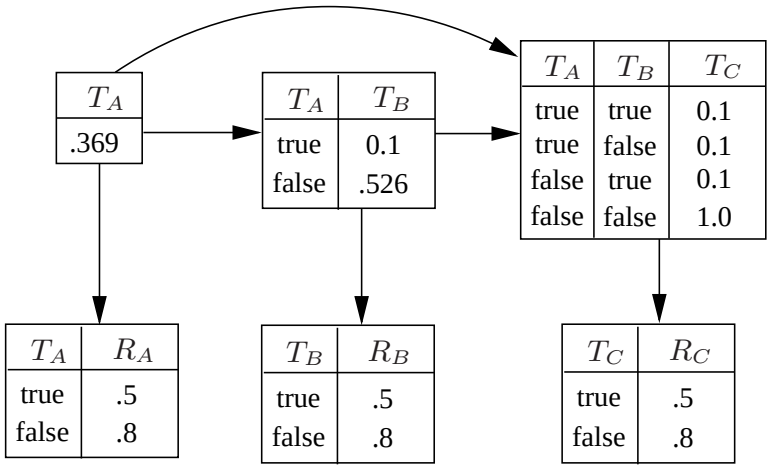


Figure 3.3: Conditional probability tables for directed clique.

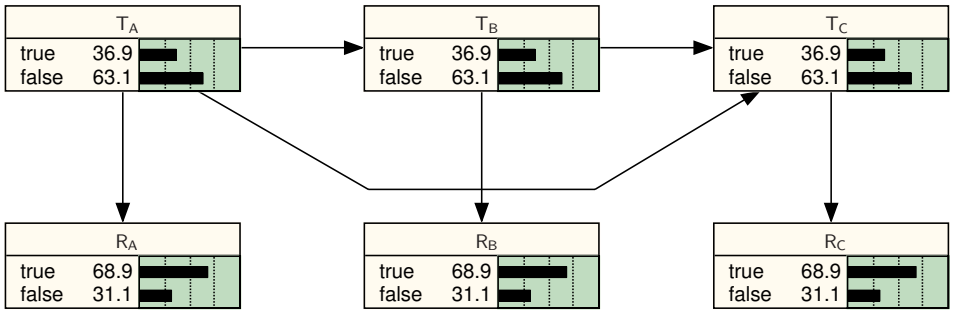


Figure 3.4: Beliefs after inference for directed clique shown in Netica. Probabilities expressed as percentages.

probabilities are expressed as percentages in Netica.

3.2 Method 2 : Chain of Sufficient Statistics

Notice in the clique method that we create ever larger tables as we move along the clique ordering. The final node will always have all other nodes in the clique as its parents. The second method is very similar to a clique but reduces the number of arcs needed by taking advantage of the fact that we do not need *all* the information, only the sufficient statistics of previous nodes. In the case of example 3 this can be expressed by binary nodes, S_1 and S_2. They need only indicate whether or not any previous T_X nodes satisfy the constraint as shown in figure 3.5.

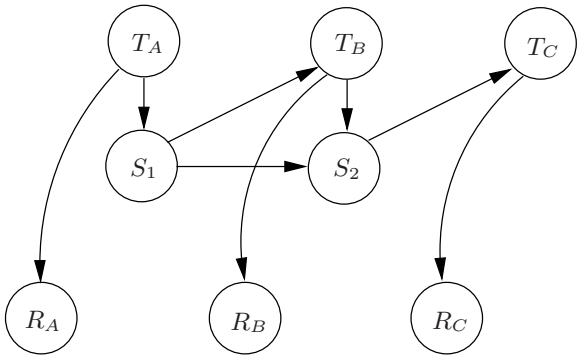


Figure 3.5: A Bayesian network using a chain of sufficient statistics to model example 3.

The T_X nodes are computed as follows:

$$p(T_A) \propto \sum_{T_B,T_C} p(T_A, T_B, T_C) \tag{3.2a}$$

$$p(T_B|S_1) \propto \sum_{T_C} p(S_1, T_B, T_C) \tag{3.2b}$$

$$p(T_C|S_2) \propto p(S_2, T_C) \tag{3.2c}$$

The sufficient statistics represent the distributions of all previous nodes in each computation.

The node S_2 summarizes the T_A and T_B nodes so that T_C only needs one input. In example 3 the resulting complexity is the same as using the clique method. However, with a larger network having more T nodes this structure only increases linearly in table size, whereas the clique method increases exponentially. Consider a network with six T_X nodes. The clique method would leave the final node in the clique with five parents and 2^6 entries. In the chain representation, however, each successive S_i node would suffice to summarize the only pertinent information, which is if any of the previous T_X nodes is true. Each T_X node would only have one parent and 2^2 entries in its table.

The computed tables using the chain method for example 3 are shown in figure 3.6. After performing inference the posterior beliefs are as shown in figure 3.7. Note that the posterior beliefs are identical to those using the clique method in 3.4.

3.3 Method 3 : Conditioning

Each of the previous methods correctly result in the distribution required by example 3 but specifying them requires carrying out inference on a portion of the network using the marginal distribution of all constrained nodes. The resulting

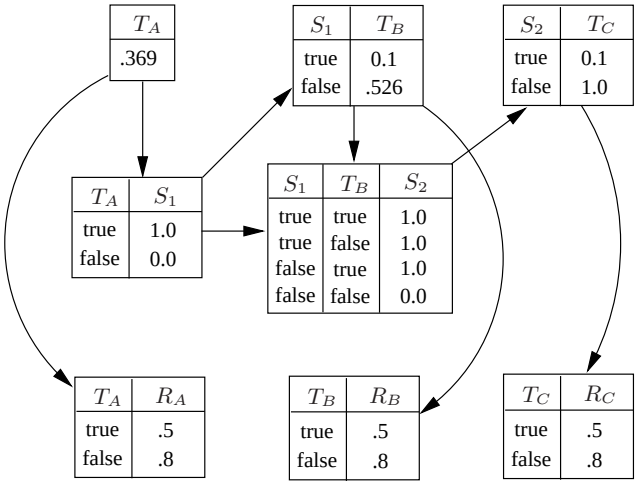


Figure 3.6: Conditional probability tables for chain method.

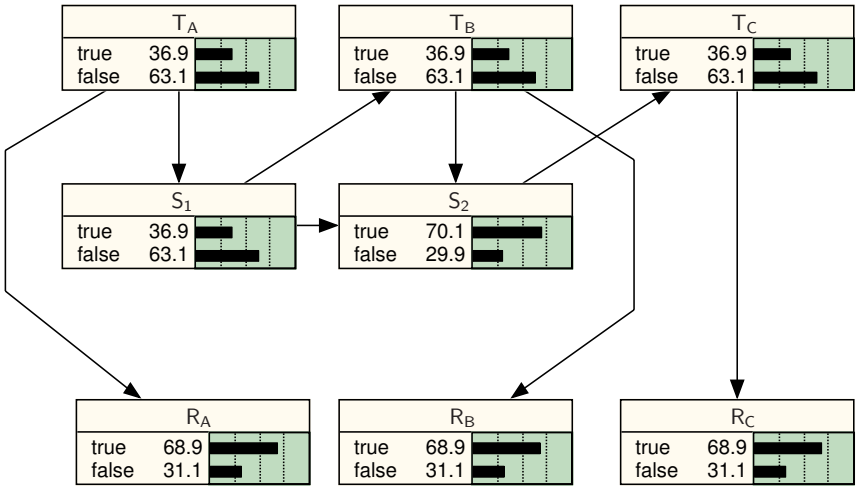


Figure 3.7: Beliefs after inference for chain shown in Netica.

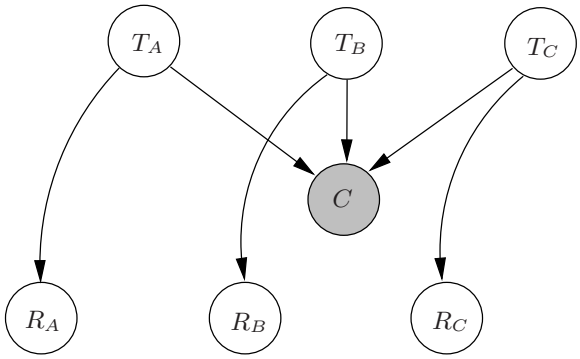


Figure 3.8: A Bayesian network using a conditioned node C to model example 3.

graph forces a direction on to an undirected constraint by allowing the influence to flow from each T_X node towards T_C so that they can become interdependent.

Conditioning provides a much simpler way to specify this distribution. A node, C, is added to the network to represent the constraint and is conditioned to be true, see figure 3.8. This entered evidence does not correspond, however, to an observation in the real world. The new node, known as a *conditioned node* or *c-node*, is true *by definition* and is part of the original specification of the joint distribution of the network. Figure 3.9 shows the CPDs for this network. These are the prior distributions given in example 3. So, $p(T_B)$ in the network is simply the $Pr(T_B)$ distribution given to us. There is no need for calculations to come up with correct tables to enter into the Bayesian network since the normal process of inference will perform the calculations of (3.1) and (3.2) automatically. This induces the required distribution on the T_X nodes as shown in figure 3.10 which behaves identically to the other two methods for example 3.

Conditioning is compelling for several reasons :

simultaneous constraint : The entire constraint is captured in one table. No artificial ordering of the variables needs to be created by the modeler. Note also that for a constraint such as "or", this conditioned node could be modelled by a more compact function instead of a table [1].

separation of priors and constraints : The two components of the distribution, directed and undirected, can be specified separately. These correspond to the prior distributions tables and the constraints in example 3. The inference mechanism deals with combining them automatically. If the modeler is given a distribution divided in this way then conditioning is completely straightforward to use.

natural modelling : Conditioning is a method that comes naturally out of the use of directed graphical models. Modelers know that entering evidence

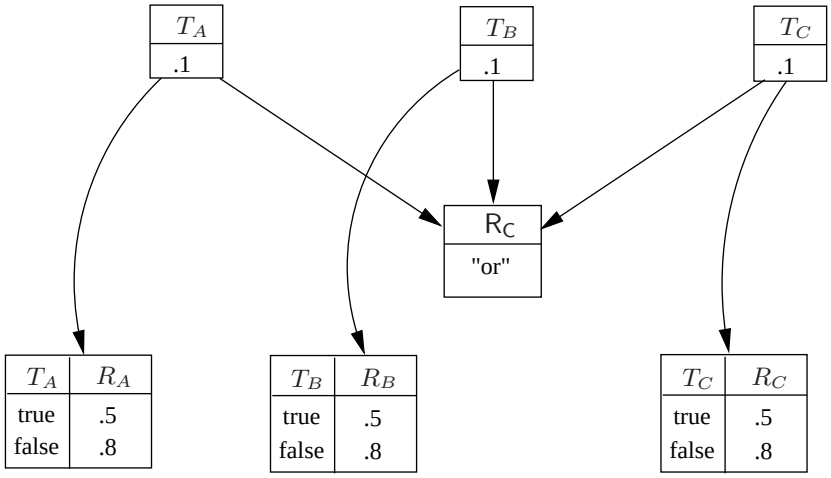


Figure 3.9: Conditional probability tables for Bayes net with conditioning.

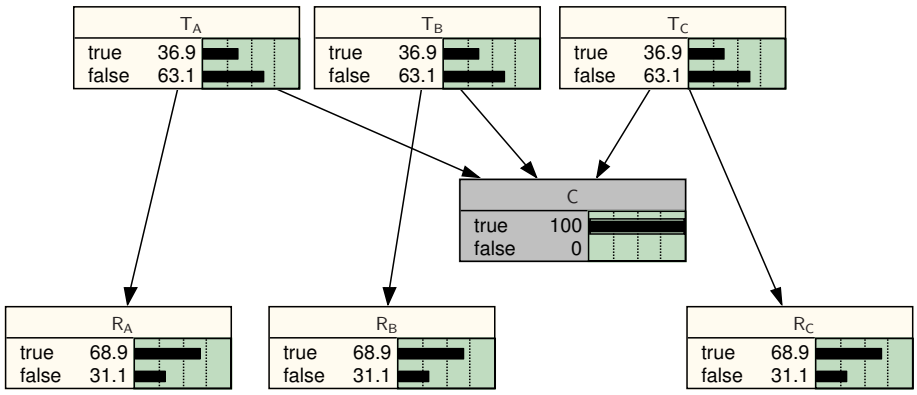


Figure 3.10: Beliefs after inference using a conditioned node to model example 3.

into a network has an effect on the parents of the conditioned node, they see it whenever an observation is made. It is natural to try to use this to more easily specify distributions that do not translate well into a directed language. No computations are needed by the modeler to specify the distribution.

Chapter 4

Side Effects of Conditioning

Now we return the ancestor variables modelling interest in AI and desire to teach from example 1 to demonstrate the problem of side effects of conditioning.

When the constrained nodes have ancestors, the use of conditioning to specify distributions creates side effects in the network. We demonstrate this using example 4.

Example 4 *Consider a model of three university instructors, Alice, Bob and Cindy, and an Artificial Intelligence (AI) course. The model has the following components:*

1. *An instructor's interest in the topic of AI: The binary variable I_X represents whether or not instructor X is interested in the topic of AI. Our belief that X is interested in teaching the course is specified as a prior distribution, $Pr(I_X)$ given by:*

X	$Pr(i_X)$	$Pr(\neg i_X)$
Alice	*.3*	*.7*
Bob	*.7*	*.3*
Cindy	*.6*	*.4*

2. *Whether an instructor wants to teach the AI course: This is modelled by the binary variable W_X for instructor X. The belief that X wants to teach the course is dependant upon their interest in the topic, I_X, represented by $Pr(W_X|I_X)$, and is the same for all three instructors:*

I_X	$Pr(w_X\|I_X)$	$Pr(\neg w_X\|I_X)$
true	*.75*	*.25*
false	*.1*	*.9*

3. *A constraint on teaching : At least one instructor must teach the course.*

4. *Whether an instructor will teach the course: This is modelled by the binary variable T_X for instructor X. Our belief that X will also be teaching the course once the constraint has been satisfied is dependant upon their desire to teach the course and is represented by, $Pr(T_X|W_X)$:*

W_X	$Pr(t_X\|W_X)$	$Pr(\neg t_X\|W_X)$
true	*.1*	*.9*
false	*.01*	*.99*

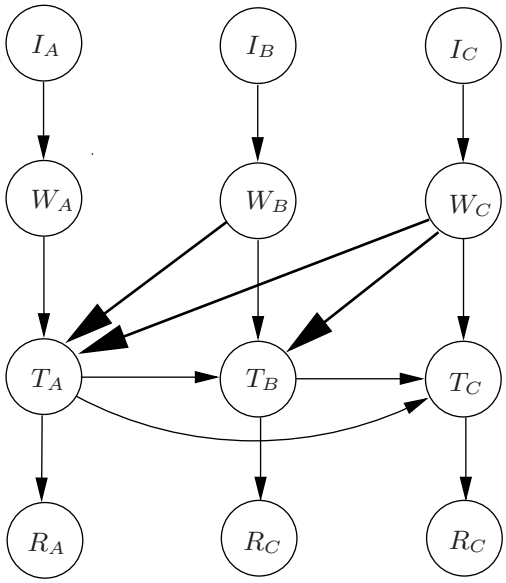


Figure 4.1: Bayesian network modelling example 1 with a directed clique.

5. *Research productivity of instructors : This is modelled by the binary variable R_X for instructor X. If R_X is true it means that instructor X will complete their research goals for this semester. Our belief in the value of R_X is dependent upon whether or not they end up teaching the AI course, T_X, which will reduce the time they can spend on research. This belief is represented by the distribution $Pr(R_X|T_X)$, and is the same for each instructor:*

T_X	$Pr(r_X\|T_X)$	$Pr(\neg r_X\|T_X)$
true	.5	.5
false	.8	.2

4.1 Modelling Ancestors with a Directed Clique

With the ancestors of the constrained T_X nodes added back in we can again model the distribution using a directed clique of nodes, shown in figure 4.1. Notice that each W_X node has a dependency connecting to its T_X node as before but there are three additional influences indicated as well. These influences, shown with bold arrows, express the fact that the distribution of each W_X node is pertinent to our beliefs about each T_X node.

To see this, consider $p(W_C)$, our belief in Cindy's desire to teach the AI course. If we have found by talking to Cindy that she does not want to teach

AI, this should reduce our belief that Cindy will teach the course, since teaching a course is influenced by the instructor's interest in the topic. This allows us to then infer that it is more likely that either Alice or Bob will teach the course since someone must teach it. However, if we think back to the Bayes ball rules described in section 2.2.1 we see that the nodes $T_B - T_C - W_C$ form a v-structure, with no observation at the bottom node, T_C. This means that effects from W_C will *not* propagate to T_B. The same is true of the effect of W_B on T_A. But if we are certain that Bob wants to teach the course then it *is* less likely that Alice or Cindy will end up teaching it. The bold arcs added to the network in figure 4.1 compensate for this problem by allowing all the ancestor nodes to influence all the constrained nodes.

The CPDs for the T_X nodes now need to be computed for the network in figure 4.1 to take the influence of these ancestors into account as we did in (3.1). Since the values of the W_X nodes now inform every combination of T_X values we determine a marginal distribution of the priors across all W_X and T_X nodes, $Pr(T_A, T_B, T_C, W_A, W_B, W_C)$. In this distribution we then enforce the constraint, assigning a probability of zero to each case where all T_X nodes are $false$. This gives us the posterior after the constraint $p(T_A, T_B, T_C, W_A, W_B, W_C)$. With this we can compute the conditional probability distributions for each T_X node as follows:

$$p(T_A|W_A, W_B) \propto \sum_{T_B, T_C, W_C} p(T_A, T_B, T_C, W_A, W_B, W_C) \tag{4.1a}$$

$$p(T_B|T_A, W_B, W_C) \propto \sum_{T_C, W_A} p(T_A, T_B, T_C, W_A, W_B, W_C) \tag{4.1b}$$

$$p(T_C|T_A, T_B, W_C) \propto \sum_{W_A, W_B} p(T_A, T_B, T_C, W_A, W_B, W_C) \tag{4.1c}$$

$$\tag{4.1d}$$

After normalizing we arrive at the distributions shown in figure 4.2 with the resulting beliefs in each node after inference shown in figure 4.3.

The chain method has a very similar specification and results in the same distribution, it is not shown here.

4.2 Modelling Ancestors with Conditioning

Figure 4.4 shows the model for example 4 using a conditioned node, C, that is a child of all nodes that need to be constrained. Again, specification of the CPDs for this network, shown in figure 4.5, is much more straightforward than the clique method. Each node for the given variables is simply assigned a CPD equal to the prior distributions provided in example 4. The CPD for C is one that matches the constraint given.

The posterior beliefs after performing inference on all nodes are shown in figure 4.6. As discussed in the Introduction conditioning in this network produces *side effects* which distort the resulting distribution. We can see the difference

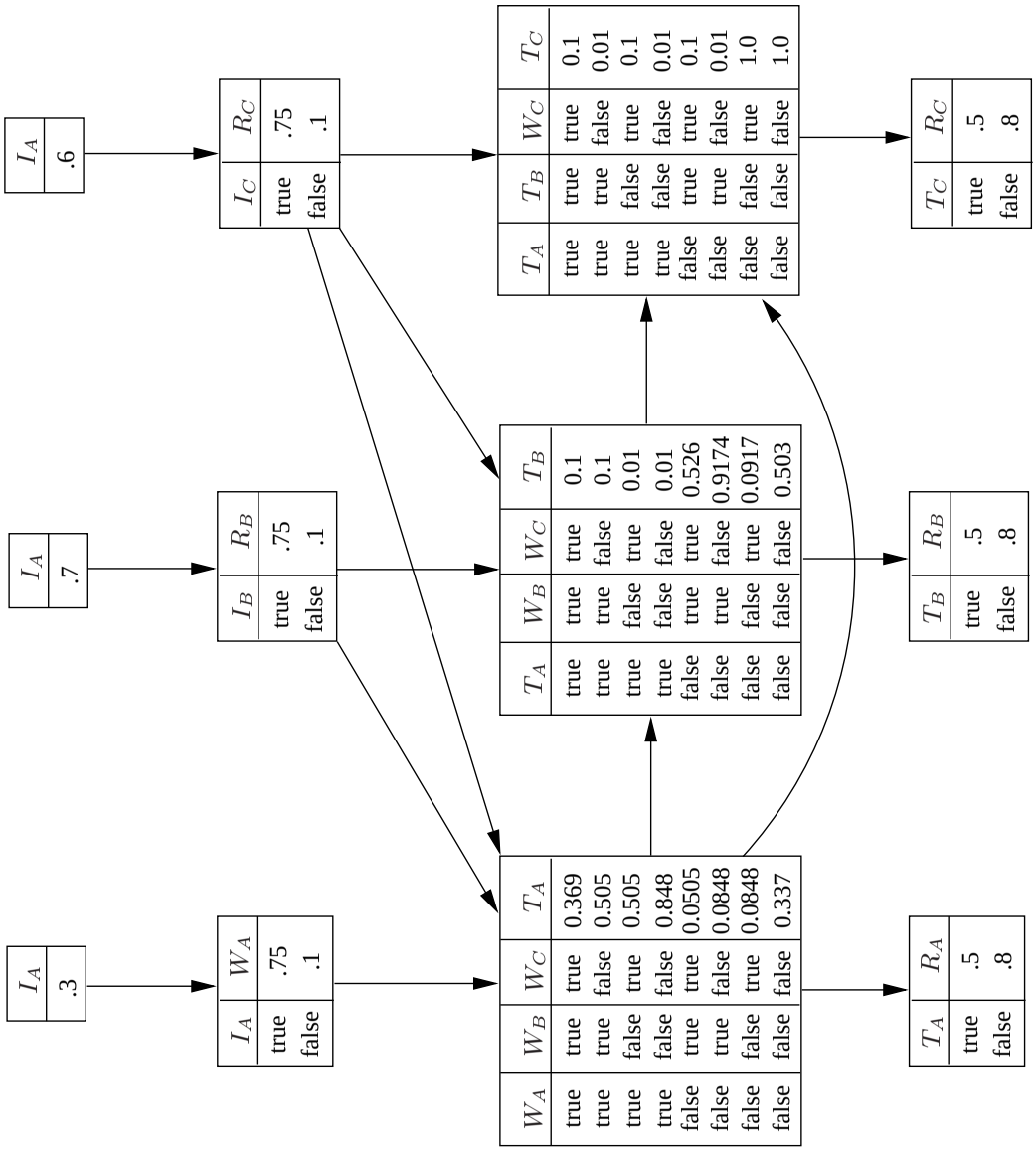


Figure 4.2: Conditional probability tables using the clique method to model example 4.

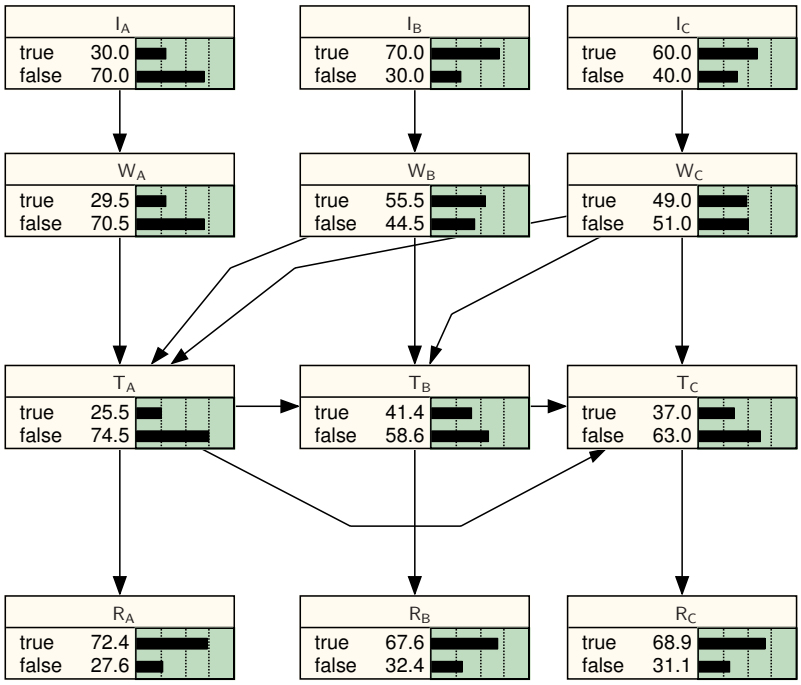


Figure 4.3: Resulting beliefs after inference is performed with a clique.

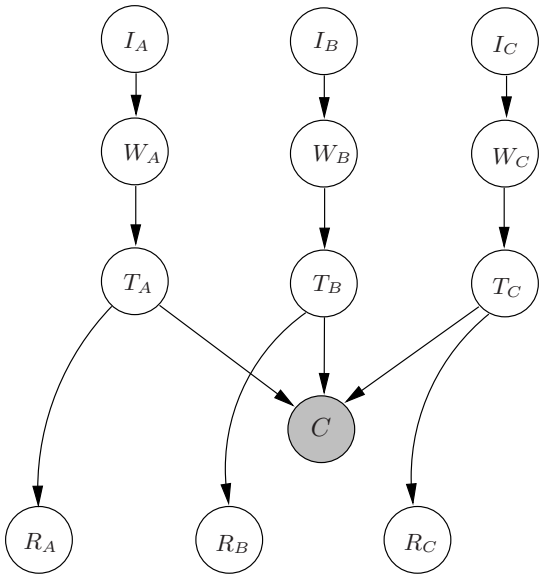


Figure 4.4: Model using a conditioned node C for example 1

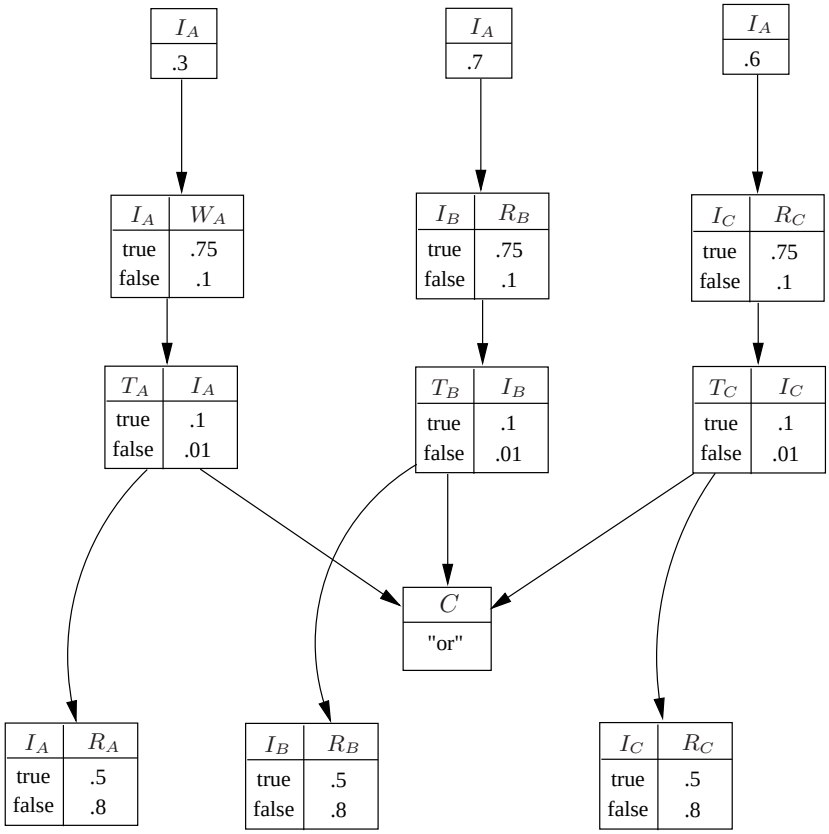


Figure 4.5: CPDs for conditioned model are simply the given prior distributions and constraint.

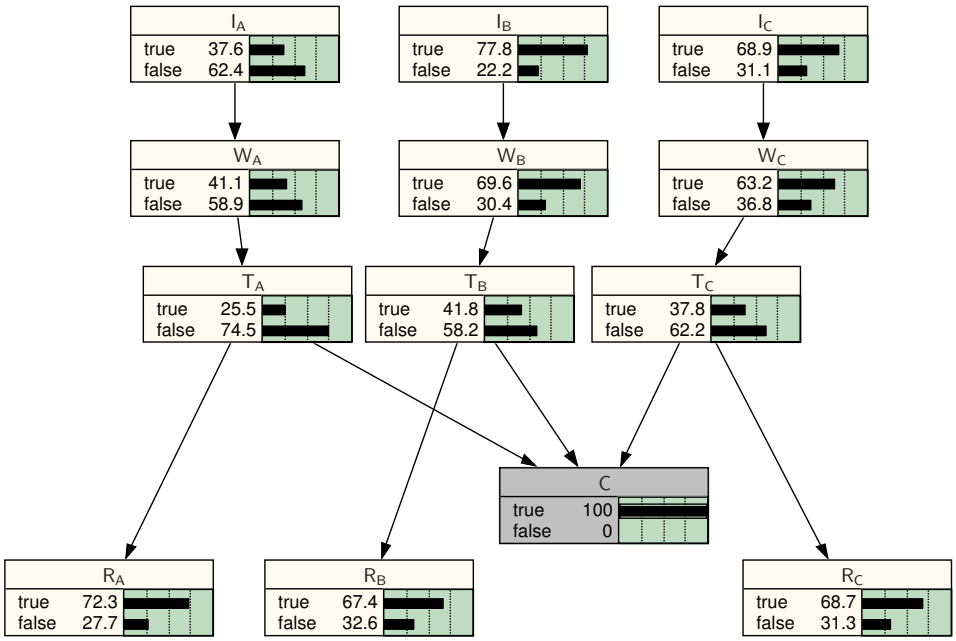
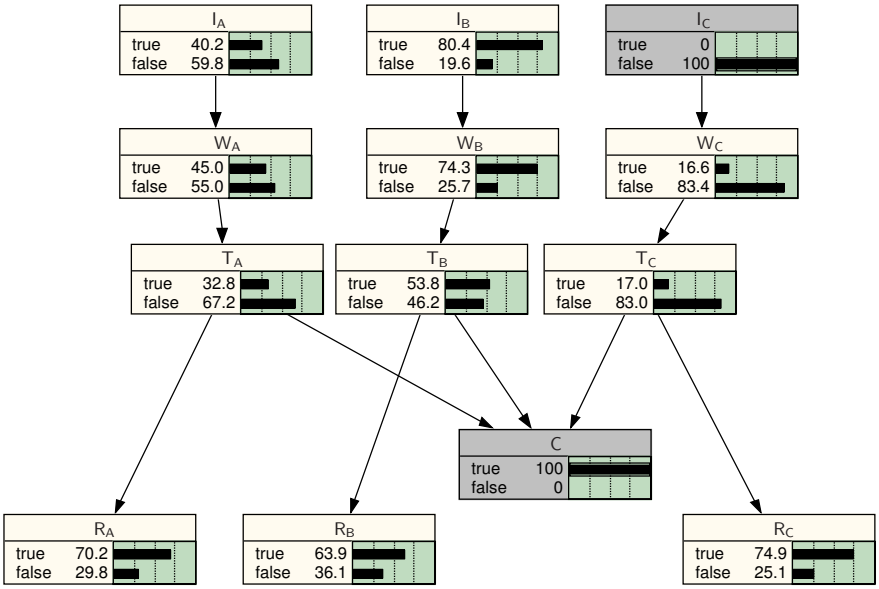


Figure 4.6: Resulting beliefs after inference is performed. Side effects of conditioning lead to a different distribution

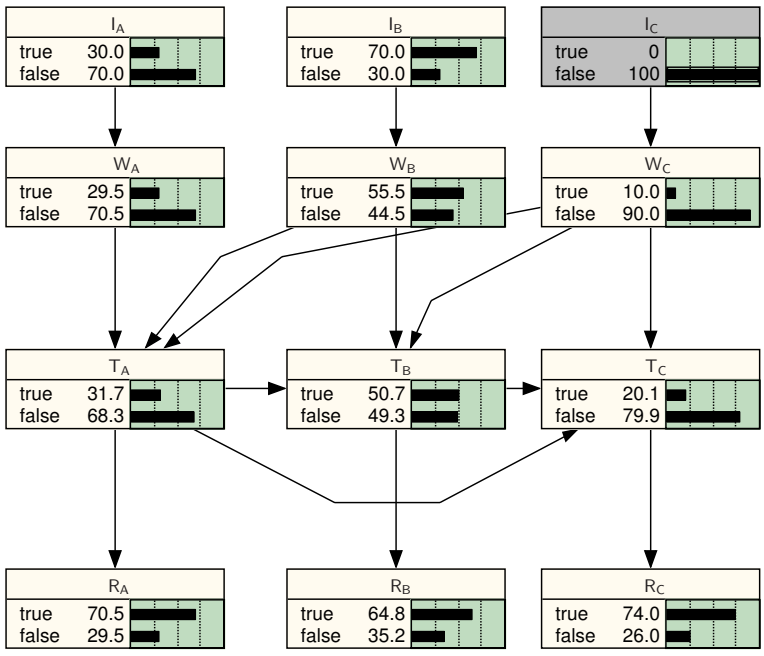
by comparing to the distribution found by the clique method, figure 4.3, where all the tables are computed without the use of evidence. If we think of the rules of 2.1 once again it is clear why this occurs. The evidence in C allows all nodes in the network to influence each other, so they all become dependent to some degree.

The problem is that we are using a mechanism, conditioning, provided by the infrastructure of Bayesian networks that is assumed to have a certain meaning, that is, observation of a variable in the model. But C is not a variable in our model. It is an expression of the undirected component of the distribution of the T_X nodes. It is straightforward to use a node in this way in a Bayes net as it makes specification much easier and yields the desired distribution across the T_X nodes. The constraint holds as expected and it may not be immediately clear that the posteriors computed by the model with conditioning are incorrect if they are not compared to other models.

Due to this subtlety it is important to create an antidote for this problem, in more complex models it will be even harder to detect. We want to stop the propagation of influence up from a conditioned node after it has affected its parents. The process of doing this will be referred to as *shielding* the I_X nodes from the side effects of the conditioning of C. This is the subject of the next chapter.



(a)



(b)

Figure 4.7: Cindy's disinterest in AI has different effects.

4.2.1 Descendants

A note on the descendants of the constrained nodes, the R_X nodes in example 4. We do not need to shield the effects of conditioning from these nodes. This is because they are defined as being influenced by our beliefs in T_X. These in turn are influenced by beliefs in W_X and I_X. So, a constraint on who can teach the course will influence our belief in the research productivity of the instructors. Determining the interest in AI of an instructor will also influence our belief in their research productivity. However, the computation of that belief yields different actual probabilities than desired. This is due to the propagation of belief amongst the I_X and W_X nodes which only occurs because of conditioning side effects.

Chapter 5

Shielding Against Side Effects

To remove the problem of side effects and maintain the useful properties of conditioning we want to devise a way to *shield* part of the network by counteracting the influence of the conditioning node. In this chapter we define some notation to aid this discussion and present a definition for what successful shielding will look like.

5.1 Conditioned, Effect and Shield Sets

In order to make it easier to refer to different parts of a network we define three sets of nodes. Each set contains one of three node types; *conditioned, affected* or *shield* nodes. Note that these are not partitions of the network. These three sets indicate what part a node plays in creating the distribution across the network, a simple network is shown in figure 5.1.

c-node A node that is created in order to induce an undirected distribution across its parents through conditioning. A *c-node* has the following properties:

- it has no children.
- it is always binary, since one value is always observed a larger domain would never be used.
- it has evidence of *true* entered into it by definition. This is what we mean by saying it is conditioned to one of its values.
- The effect of a *c-node* is intended to be local to its parents and by extension to all of its descendants, no other nodes should be influenced by the conditioning. It represents the undirected or constrained distribution across its parent nodes.

A given Bayesian network can contain multiple *c-nodes*.

e-node A *c-node* produces a direct *effect* on this node. This is simply the set of parents of the *c-node* , $pa(C_i)$ for a *c-node* C_i. We define the effect set for each *c-node* as all the nodes affected by it, that is, $\mathbf{E}_{C_i} = \{E^1_{C_i}, \ldots, E^m_{C_i}\}$ where C_i has m parents.

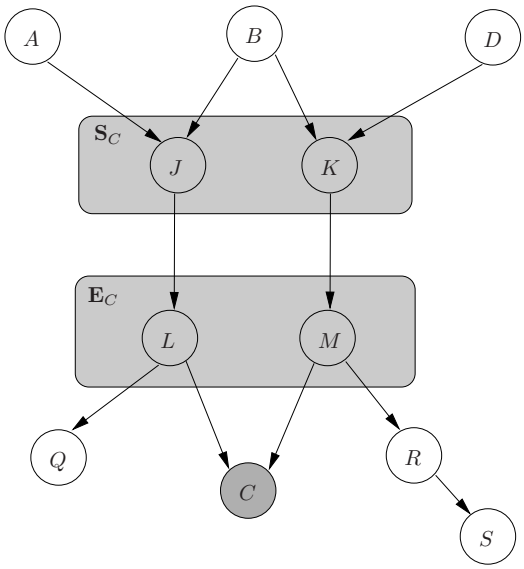


Figure 5.1: Shield and effect sets of nodes are: $\mathbf{S_C} = \{J, K\}$, $\mathbf{E_C} = \{L, M\}$

s-node A *shield* node is the first line of defence against the propagation of side effects. We define a set $\mathbf{S}_{C_i}$ for each *c-node*, C_i, as all nodes in the set of "grandparents" of C_i that are not also an affected node of C_i, that is,

$$\mathbf{S}_{C_i} = pa(\mathbf{E}_{C_i}) - \mathbf{E}_{C_i}.$$

We exclude $\mathbf{E}_{C_i}$ since it is possible that linking between *e-nodes* could cause some *e-nodes* to be grandparents of C_i. . If there are k of these nodes then the members of the set are $\mathbf{S}_{C_i} = \{S^1_{C_i}, \ldots, S^k_{C_i}\}$.

To be shielded from the side effects of C means that after inference the following holds for each *s-node* :

$$p(S^j_C | C = true) = p(S^j_C)$$

That is, in the absence of any other evidence, S^j_C should behave as if the node C were not conditioned.

In a network containing *c-nodes*, the specification of the network is not complete until all the *c-nodes* are conditioned to true and some method has been introduced to shield the nodes in $\mathbf{S_C}$ that should not be affected by this conditioning.

5.1.1 Affected Network

We will also refer to the portion of the network that the *c-node* affects as the *affected network*. This is all of the nodes that are influenced by the conditioning

of C and the node C itself. It follows from our previous definitions that this set contains all the *e-nodes* and their descendants, which are not shielded.

$$\mathbf{E}_C \cup desc(\mathbf{E}_C) \tag{5.1}$$

5.2 Defining Shielding

Our goal now is to implement some unobstructive method of shielding. By unobstructive we mean that it is a solution that requires no modification of standard Bayes net inference algorithms. Ideally, a modelling tool allowing specification of distributions using *c-nodes* would be able to implement some method to shield conditioning side effects without the modeler needing to know how it was implemented in their network. The solution we describe can be entered into any existing Bayesian network modelling package to produce the desired distribution.

5.2.1 Anti-nodes

We begin with the idea that what conditioning has wrought, perhaps it can undo as well. Figure 5.2 shows a simple network with two *anti-nodes*, $\hat{A}_1$ and $\hat{A}_2$, one for each *s-node* to be shielded. We will use this "hat" notation for all nodes added to counter side effects. The CPD of each anti-node is computed so that it shields its parent from the side effects of C, that is, the following holds:

$$p(J|c, \hat{a}_1, \hat{a}_2) = p(J) \tag{5.2a}$$
$$p(K|c, \hat{a}_1, \hat{a}_2) = p(K) \tag{5.2b}$$

Although this can maintain the individual distributions $p(J)$ and $p(K)$ in this network they are unable to deal with more complex distributions. Without conditioning we have that

$$p(J|K) = p(J)$$

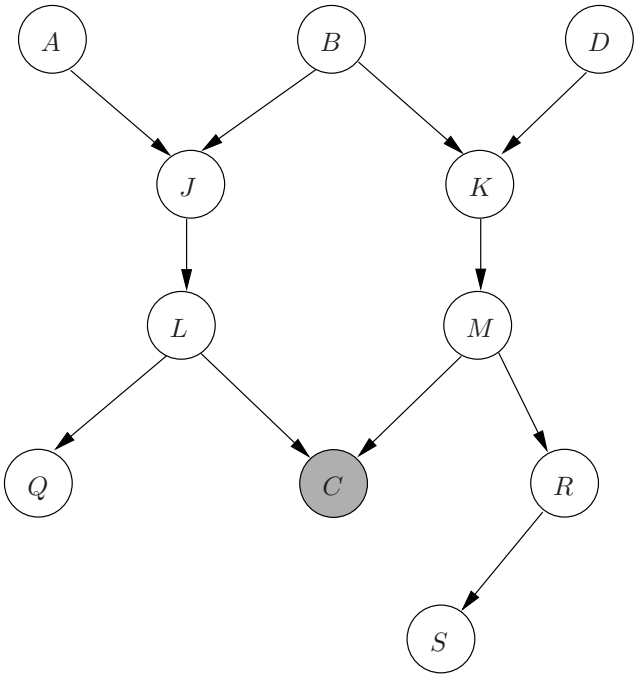
but with conditioning and the anti-nodes

$$p(J|K, c, \hat{a}_1, \hat{a}_2) \neq p(J|c, a_1, a_2).$$

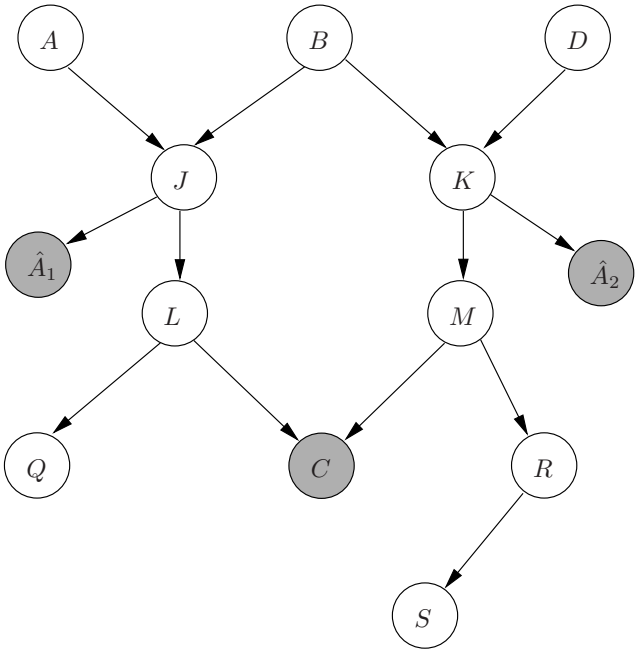
Anti-nodes, then, are not connected enough to compensate for these effects. What we need is an expression that encompasses the total influence of the nodes in the affected network.

5.2.2 Anti-factors

Side effects are an influence on the joint distribution of the *s-nodes*. So, to counter this influence we create another conditioned node, $\hat{C}$ with parents $\mathbf{S_C} = \{J, K\}$, see figure 5.3, and determine if there is a CPD that will exactly counter C's effects. Consider how inference will proceed using variable elimination:



(a) A basic example of a conditioned network.



(b) Network with antinodes added.

Figure 5.2: A first try at shielding with anti-nodes.

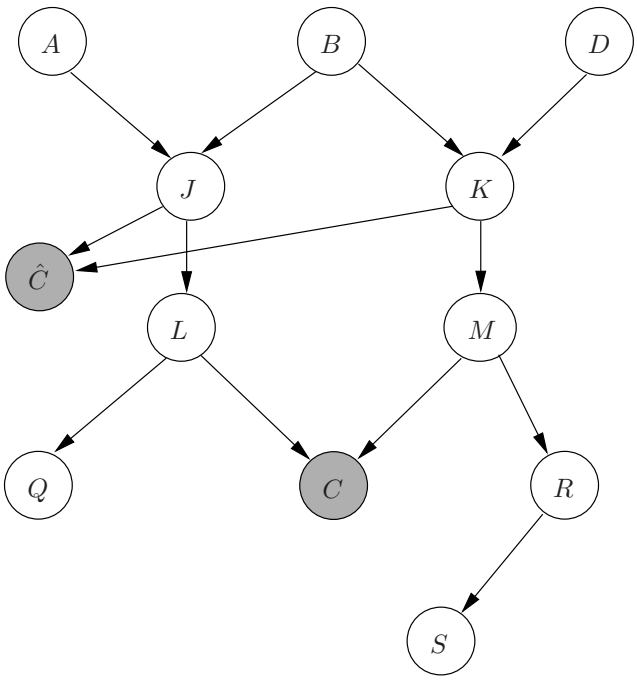


Figure 5.3: A factor produced for a *c-node* has a corresponding *anti-factor* on its *s-nodes*.

$$
\begin{aligned}
p(J,K,c,\hat{c}) &= \sum_{A,B,D,L,M} p(J|A,B)p(A)p(B)p(K|B,D)p(D) \\
&\qquad \times p(\hat{c}|J,K)p(L|J)p(M|K)p(c|L,M) \\
&= \sum_{A,B,D} \mathbf{p(J|A,B)p(A)p(B)p(K|B,D)p(D)} \\
&\qquad \times p(\hat{c}|J,K) f_{L,M}(J,K)
\end{aligned} \tag{5.3}
$$

If the affected network is eliminated first, the factor $f_{L,M}(J,K)$ is produced leaving us with the unknown distribution of the $\hat{C}$ node and the remainder of the network shown in bold. This remaining portion is simply $p(J,K)$. We assign the CPD for $\hat{C}$ then as

$$
\begin{aligned}
p(\hat{c}|J,K) &= \frac{1}{Z}\frac{1}{f_{L,M}(J,K)} \\
p(\neg\hat{c}|J,K) &= 1 - \frac{1}{Z}\frac{1}{f_{L,M}(J,K)}
\end{aligned}
$$

where $Z = \sum_{J,K} f_{L,M}(J,K)$ is a normalization constant that ensures these values are in the range [0,1]. The node $\hat{C}$, called an *anti-factor*, when conditioned to *true*, will cancel out with the factor $f_{L,M}(J,K)$. During inference this will yield:

$$p(J,K,c,\hat{c}) = p(J,K)$$

General Anti-factors

In general, we create an anti-factor node, $\hat{C}$, with parents $\mathbf{S_C}$ such that

$$
\begin{aligned}
p(\hat{c}|\mathbf{S_C}) &= \frac{1}{Z}\frac{1}{f_{\mathbf{E_C}}(\mathbf{S_C})} \\
p(\neg\hat{c}|\mathbf{S_C}) &= 1 - \frac{1}{Z}\frac{1}{f_{\mathbf{E_C}}(\mathbf{S_C})}
\end{aligned}
$$

where $Z = \sum_{\mathbf{E_C}} f_{\mathbf{E_C}}(\mathbf{S_C})$. With $\hat{C}$ conditioned to *true* the following is the definition of shielding:

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

Instructor Example

Figure 5.4 shows the CPD computed for the anti-factor in our instructor example. The values come from the antifactor of the affected network given by

$$\hat{f}_{T_A,T_B,T_C}(I_A,I_B,I_C) = \frac{1}{Z}\frac{1}{f_{T_A,T_B,T_C}(I_A,I_B,I_C)}. \tag{5.4}$$

J	$p(l = \mathbf{t}\|J)$
t	.59
f	0

(a)

K	$p(m = \mathbf{t}\|K)$
t	0
f	.33

(b)

L	M	$p(c = \mathbf{t}\|L, M)$
t	**t**	1
f	**t**	1
t	**f**	1
f	**f**	0

(c)

J	K	$f_{L,M}(J, K)$
t	**t**	.59
f	**t**	0
t	**f**	.7194
f	**f**	.33

(d)

Table 5.1: CPDs can lead to a factor containing zeros.

We can see the resulting posterior distribution in figure 5.5. Notice that the this matches exactly with the distribution shown using the clique method in figure 4.3.

Inverting Zero

There appears to be a problem with the computation of an anti-factor, we cannot invert a factor containing a zero. For example, consider the network from figure 5.3 using the CPDs from table 5.1.a-c.

Using these CPDs to compute the factor resulting from eliminating L and M we arrive at the result shown in 5.1(d). The computation for the case resulting in zero is shown below:

$$\begin{aligned} f_{L,M}(\neg j, k) &= \sum_{L,M} p(L|\neg j)p(M|k)p(c|L, M) \\ &= p(l|\neg j)p(m|k)p(c|l, m) + p(\neg l|\neg j)p(m|k)p(c|\neg l, m) \\ &\quad + p(l|\neg j)p(\neg m|k)p(c|l, \neg m) + p(\neg l|\neg j)p(\neg m|k)p(c|\neg l, \neg m) \\ &= 0 \times 0 \times 1 + 1 \times 0 \times 1 + 0 \times 1 \times 1 + 1 \times 1 \times 0 \\ &= 0 \end{aligned}$$

Theorem 1 *For a factor $f_{\mathbf{E_C}}$ to contain zeroes it is sufficient that for each $\mathbf{e_c} \in dom(\mathbf{E_C})$*

- *some node $x \in \mathbf{e_C}$ has $p(x|\mathbf{S_C}) = 0$*
- *or $p(c|\mathbf{e_C}) = 0$*

Furthermore, it is necessary that for some $\mathbf{e_C}$ it is the case that $p(c|\mathbf{e_C}) = 0$.

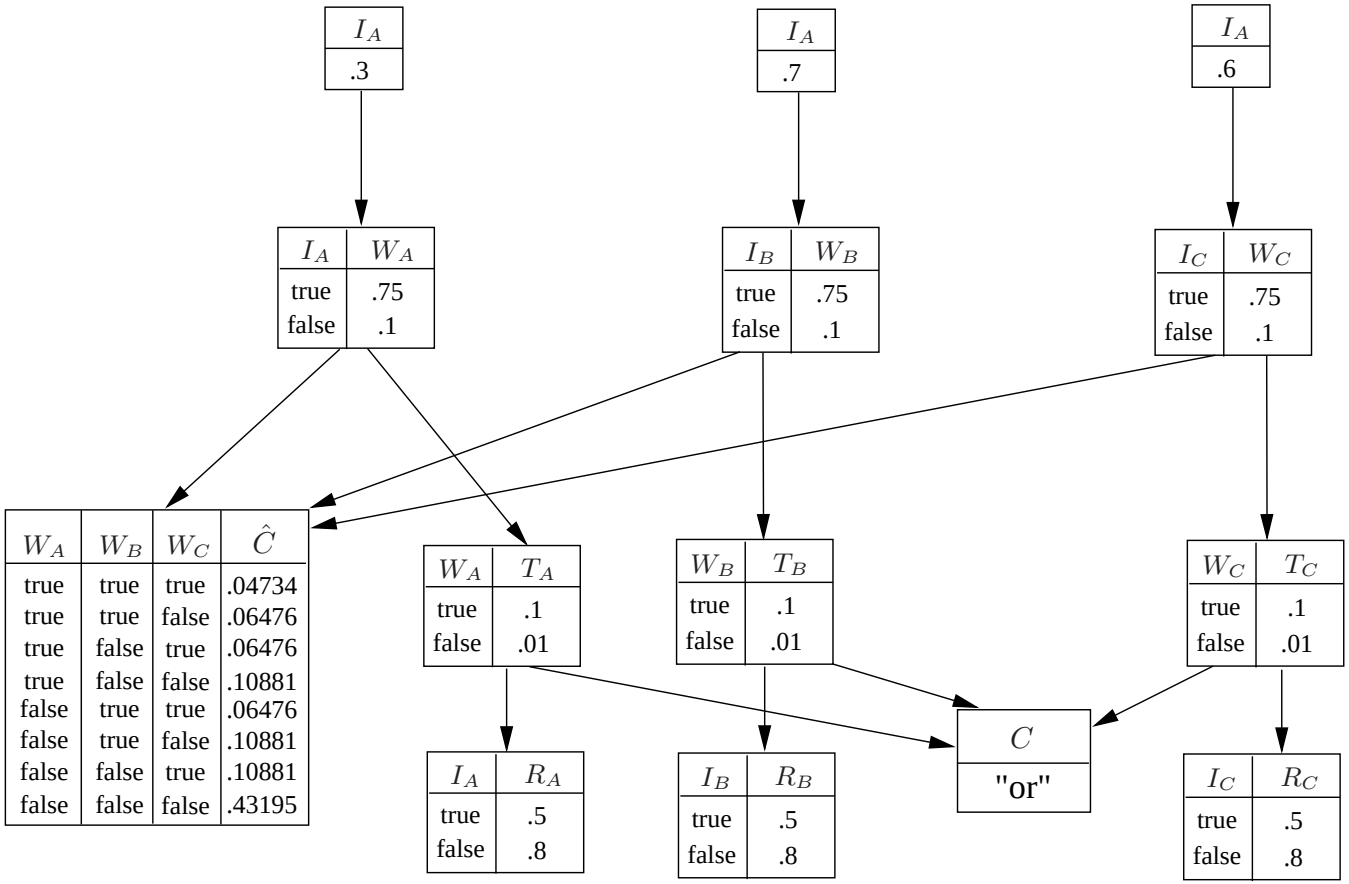


Figure 5.4: CPD for the antifactor $\hat{C}$ for example 1

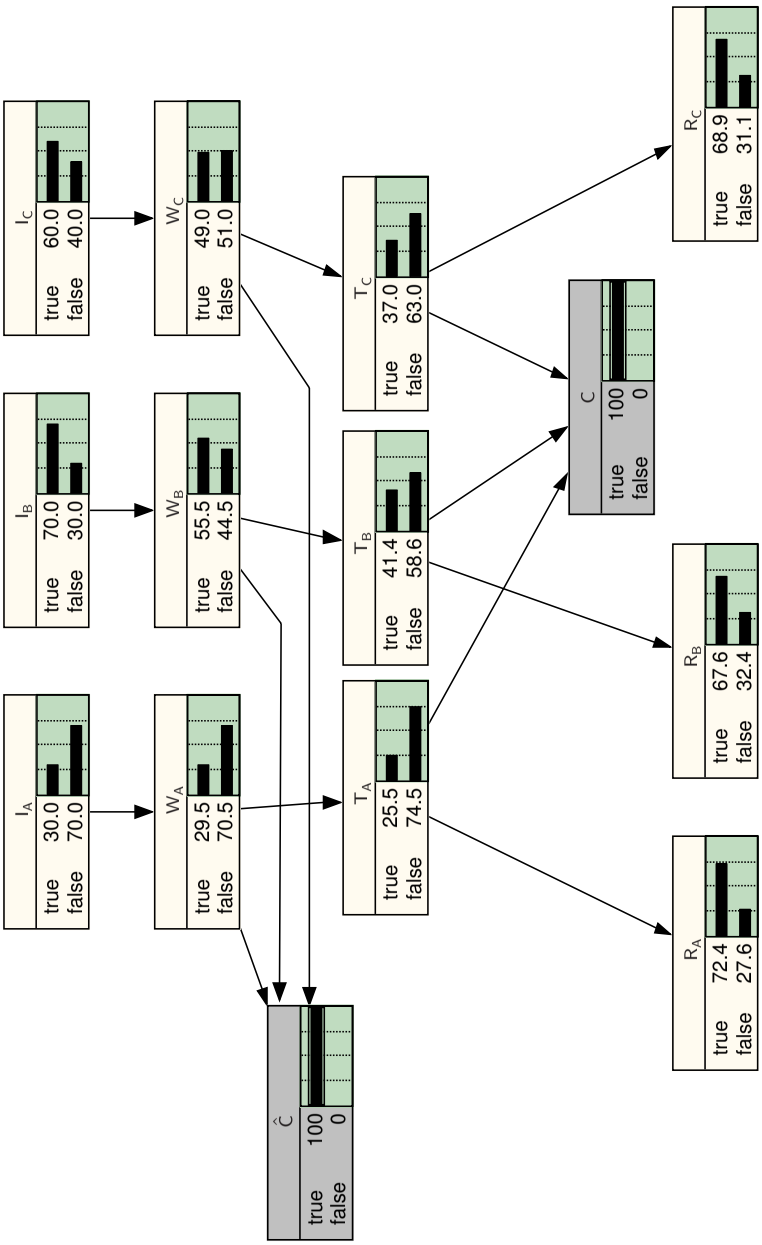


Figure 5.5: Posterior distributions after inference with conditioning and a shielding anti-factor node $\hat{C}$ for example 1.

Proof: We can see from the form of the computation for the factor that the first two cases are sufficient to make the final answer zero.

$$f_{\mathbf{E_C}}(\mathbf{S_C}) = \sum_{\mathbf{e_c} \in dom(\mathbf{E_C})} p(c|\mathbf{e_c}) \prod_{x \in \mathbf{e_c}} p(x|\mathbf{S_C}) \tag{5.5}$$

As long as one of the probabilities is always zero then each term in the sum will be zero.

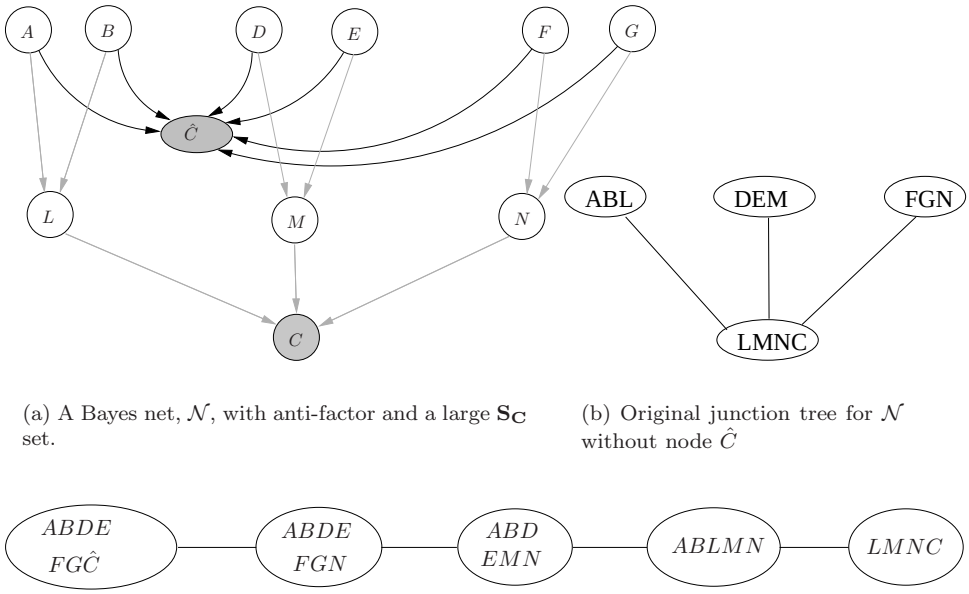
To see that it is also necessary for the *c-node* to be zero at some point, consider if it were not so. Assume $p(c|\mathbf{e_c})$ never equals zero for any instantiation of $\mathbf{e_c}$, yet some entry in the factor is still zero. Since $p(x|\mathbf{S_C})$ is a probability, there are no negative values and the only way for the sum to equal zero is for each and every term to be zero. For $\mathbf{E_C} = \{E_C^1, \ldots, E_C^n\}$, all binary nodes, there are 2^n instances of $\mathbf{e_c}$. If each node has probability zero for one of its values it must have probability one for the other value. There will still be one instance of $\mathbf{e_c}$ where every $p(x|\mathbf{S_C})$ term has a probability of one. For this instance the only way to obtain a zero would be to have $p(c|\mathbf{e_c}) = 0$. Which breaks our assumption. Therefore there cannot be a zero in the factor without $p(c|\mathbf{e_c}) = 0$ at some point. □

The importance of this situation should not be overstated. The meaning of a factor containing a zero is that for some instance of the values of the parents, the *s-nodes* in our case, the combined probability of the affected network for that instance is zero. The affected network, the combined distribution of the *e-nodes* and their ancestors including the conditioned node, is basically saying this assignment of values to the *s-nodes* is impossible. By trying to shield the side effects in this case we are trying to say that the impossible is possible. Since the only way we have of canceling out a probabilistic influence is through scaling, we simply cannot do it in this situation, nor should we try. The affected network has essentially imposed a hard constraint directly onto the values of $\mathbf{S}_C$ and the multiplication by zero has destroyed information that cannot be recovered.

Complexity

Simple and effective as anti-factors are, they do have a drawback. We are creating one new node to represent a factor of many nodes. Thus it is possible that the node $\hat{C}$ will have a much higher inwards arity than any other node in the network, see figure 5.6. If we think of inference using the junction tree algorithm what happens is that now there is new clique in the network containing the anti-factor node and *all* of the *s-nodes*. There was not necessarily a clique this large previously. Since inference is exponential in the size of the largest clique in the junction tree this method could easily lead to large increases in computation time arbitrarily influenced by the size of $\mathbf{S_C}$.

In the next chapter we propose another solution, *cloned anti-networks*, that will get around this difficulty while retaining the shielding properties of the anti-factor.



(a) A Bayes net, $\mathcal{N}$, with anti-factor and a large $\mathbf{S_C}$ set.

(b) Original junction tree for $\mathcal{N}$ without node $\hat{C}$

(c) Junction tree for $\mathcal{N}$

Figure 5.6: Antifactors can cause significant increase in the size of cliques in the junction tree if $\mathbf{S_C}$ is large.

Chapter 6

Cloned Anti-networks

To reduce the size of the clique created by the anti-factor we create instead a conditional structure in the network that has the same effect as a single anti-factor node. The most straightforward way to do this is to duplicate the existing conditional structure of the affected network and somehow compute the CPDs for these nodes so that the factor produced equals the anti-factor. This way, the shielding will still take place without creating a new clique arbitrarily larger than any existing clique.

6.1 Definition

Given a Bayesian network with *c-node* C and sets $\mathbf{E_C}$ and $\mathbf{S_C}$ defined in the usual way, we define the *cloned anti-network* for C as the set of nodes

$$\hat{\mathbf{E}}_\mathbf{C} = \{\hat{E}^i_\mathbf{C} | \mathrm{dom}(\hat{E}^i_\mathbf{C}) = \mathrm{dom}(E^i_\mathbf{C}), pa(\hat{E}^i_\mathbf{C}) = pa(E^i_\mathbf{C})\}$$

plus the node $\hat{C}$ satisfying $\mathrm{dom}(\hat{C}) = \mathrm{dom}(C)$ and $pa(\hat{C}) = pa(C)$, see figure 6.1.

The nodes in the anti-network are referred to as the *clones* of their counterparts in the original network.

The requirement for a solution is that the CPDs of the nodes in $\hat{E}_\mathbf{C}$ and $\hat{\mathbf{C}}$ are assigned such that they satisfy the following:

$$\begin{aligned} f_{\hat{\mathbf{E}}_\mathbf{C}}(\mathbf{S_C}) &= \sum_{\hat{E}^1_\mathbf{C},\ldots,\hat{E}^n_\mathbf{C}} p(\hat{E}^1_\mathbf{C}|pa(\hat{E}^1_\mathbf{C})),\ldots,p(\hat{E}^n_\mathbf{C}|pa(\hat{E}^n_\mathbf{C}))p(c|\hat{E}^1_\mathbf{C},\ldots,\hat{E}^n_\mathbf{C}) \\ &\propto \frac{1}{\sum_{E^1_\mathbf{C},\ldots,E^n_\mathbf{C}} p(E^1_\mathbf{C}|pa(E^1_\mathbf{C})),\ldots,p(E^n_\mathbf{C}|pa(E^n_\mathbf{C}))p(c|E^1_\mathbf{C},\ldots,E^n_\mathbf{C})} \\ &\propto \frac{1}{f_{\mathbf{E_C}}}(\mathbf{S_C}) \end{aligned} \tag{6.1}$$

Ensuring this is not as straightforward as in the anti-factor case unfortunately. We need to specify the conditional probabilities of an entire subnetwork so that when the affected nodes are eliminated they yield a particular, known factor. In a sense we are performing variable elimination in reverse. We already know the factor but need to set the CPDs in a such a way that we arrive at it.

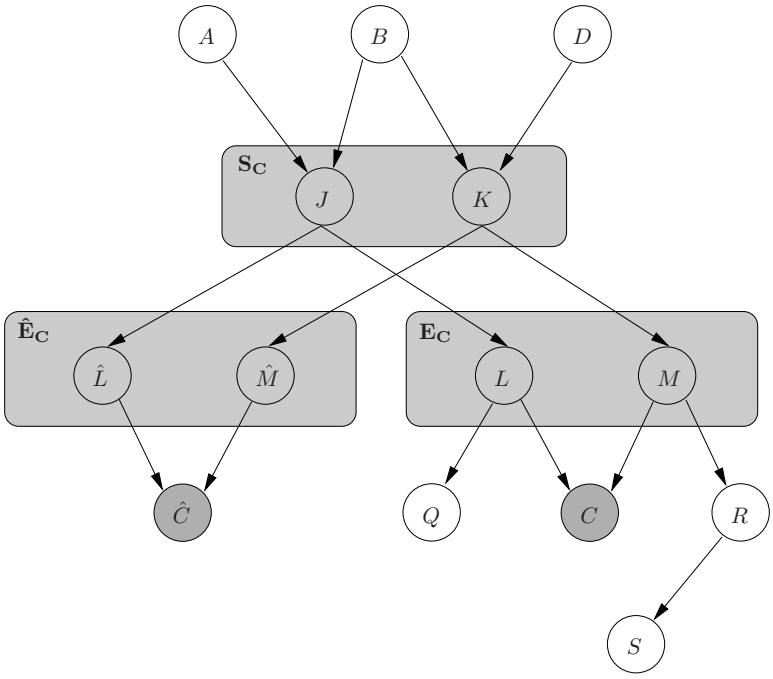
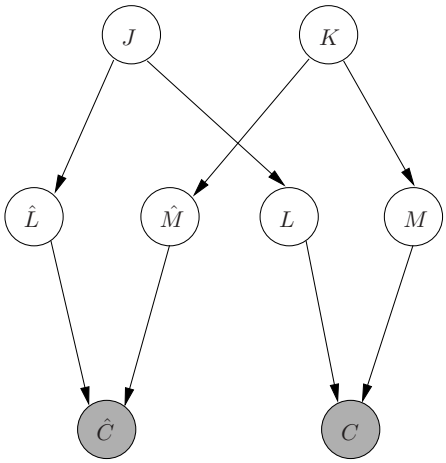


Figure 6.1: Effect sets in a Bayesian network with an anti-network.

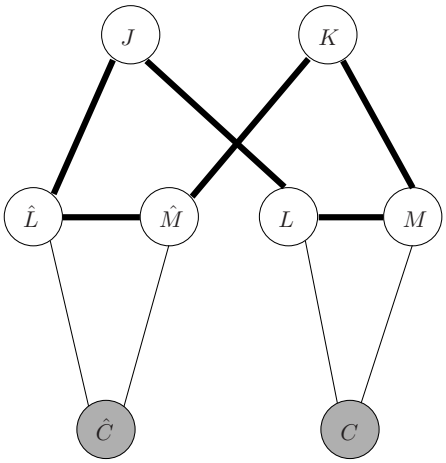
6.2 Junction Tree Complexity

Consider the network with maximum independence among the *s-nodes*, figure 6.2(a) and how it will be triangulated. After moralization, see figure 6.2(b), there is a cycle of length 6. Moralization will always link the nodes in $\mathbf{E}_C$ to each other and the nodes in $\hat{\mathbf{E}}_C$ to each other in this way. *E-nodes* also always share a parent with their copy in the anti-network. Therefore, these kinds of cycles will always arise unless no *e-nodes* have any parents, in which case shielding is unnecessary. In order to fully triangulate the graph we can cut the route short for each *e-node* by linking it to its clones in the anti-network, as in figure 6.2(c).

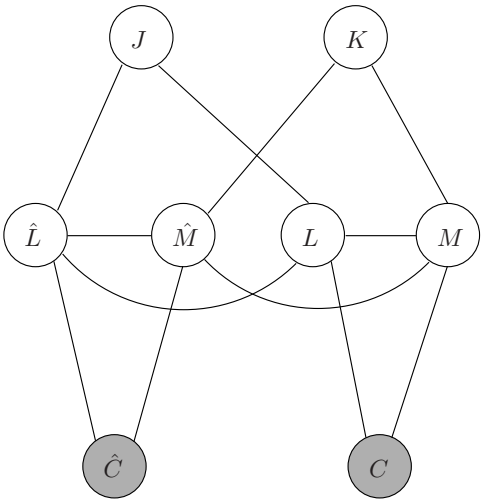
A more complex example is shown in figure 6.3. The links between an *e-node* and its clone will ensure, as in this example, that occurrences of an *e-node* in the original junction tree will now also contain the anti-network clone of that node. In the worst case this will create a clique that is twice the size of cliques in the tree without an anti-network. Consider however that if an anti-factor is used instead then an even larger clique would be created in this example. An anti-factor as a child of all *s-nodes* would create a clique of size seven. In some networks there could be very low connectivity between the *s-nodes* before shielding. After shielding with an antifactor, all *s-nodes* would be linked and form a clique together with the anti-node. Linking all of the *s-*



(a) The original Bayesian network



(b) Network after moralization has a cycle of length 6.



(c) Joining *e-nodes* to their anti-network counterparts removes the cycles.

Figure 6.2: Adding an anti-network and its effects on triangulation of the network.

nodes together is the simplest way to be sure all effects can be compensated for. However, using anti-networks, cliques will only be created where needed due to the dependencies amongst the *s-nodes*. Therefore, depending on the network structure, anti-networks could provide significant complexity advantages over the use of anti-factors.

6.3 A General Solution

In this section we present how to populate the CPDs of the nodes in the anti-network in order to enable shielding as we have described. We begin with a simple example network showing what is known about the nature and existence of a solution. We will then derive a general form applicable to any network and show the same results. Implementation details of the solution, using constrained, nonlinear, optimization and resulting networks are discussed in chapter 7.

6.3.1 Base Example

Consider the example network shown in figure 6.4, all nodes are binary. We use the concept of a cloned anti-network to cancel out the side effects of C in the portion of the network including and above $S_C = \{J, K\}$, the shield set. The parameters of the anti-network need to be set so that the factor produced after elimination of its nodes during inference conforms to:

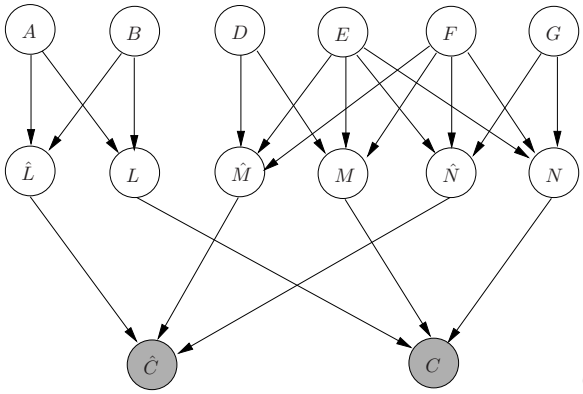
$$\hat{f}_{\hat{L},\hat{M}}(J,K) = \frac{1}{Z}\frac{1}{f_{L,M}(J,K)} \tag{6.2}$$

Where $Z = \sum_{J,K} f_{L,M}(J,K)$ is the normalization constant used to keep the anti-factor entries in the range $[0,1]$.

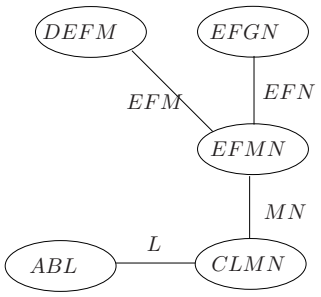
To deal with the parameters more compactly we will adopt the following notation. The free parameters of the system are $p(\hat{C} = true|L,M)$, $p(\hat{L} = true|J)$ and $p(\hat{M} = true|K)$. These will be represented by the variables γ_{lm}, ϕ_j and ψ_k respectively. The subscripts of these variables indicate the values of their parent nodes with 1 indicating true and 2 indicating false with the parents ordered from left to right in the graph. The set of all these variables is $\mathbf{X} = \{\gamma_{11}, \gamma_{12}, \gamma_{21}, \gamma_{22}, \phi_1, \phi_2, \psi_1, \psi_2\}$. The values of the factor for the anti-network, given by equation (6.2), which are constants, will be represented in the same manner by π_{jk}. Since all the variables in $\mathbf{X}$ represent probabilities, $dom(x) = [0,1]$ for all $x \in \mathbf{X}$. Similarly, $dom(\pi_{jk}) = [0,1]$ for all j,k due to Z in equation (6.2).

Thus the table for $\hat{C}$ is

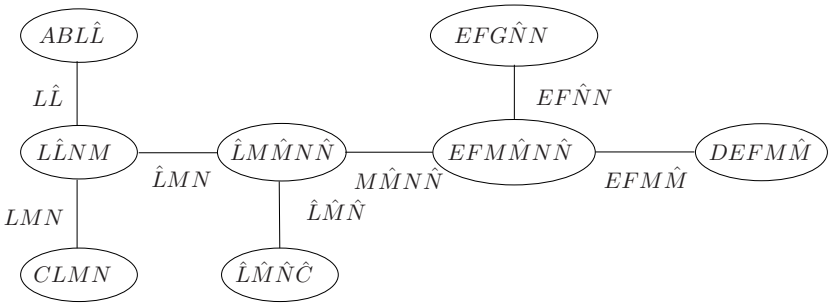
$\hat{L}$	$\hat{M}$	$p(\hat{c}\|\hat{L},\hat{M})$
t	t	γ_{11}
t	f	γ_{12}
f	t	γ_{21}
f	f	γ_{22}



(a) Bayesian network with multiple connections between *e-nodes* and *s-nodes*.



(b) Junction tree network if the anti-network were not present.



(c) Junction tree for network with anti-network present.

Figure 6.3: A more complex network showing how connectivity between *e-nodes* and *s-nodes* can lead to clique almost twice as large as those present without the anti-network.

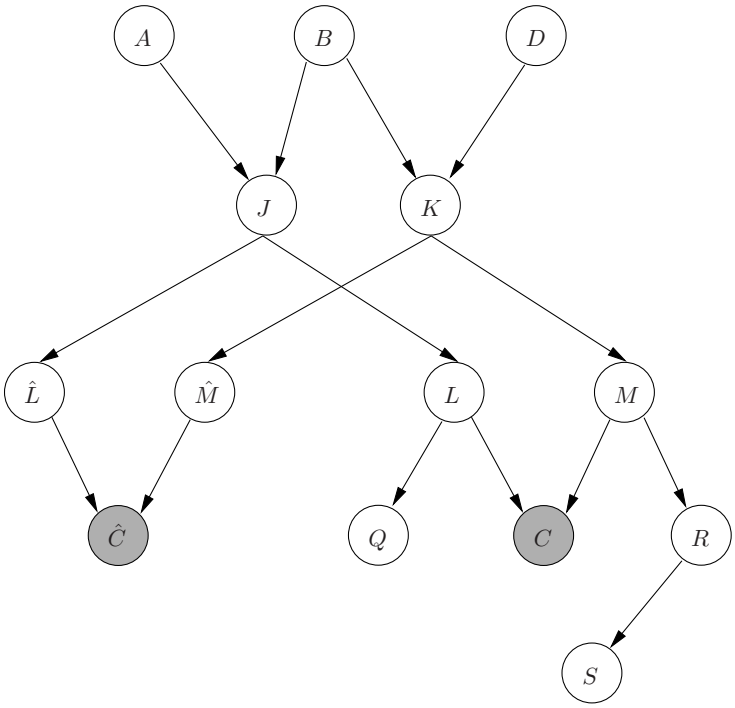


Figure 6.4: Bayesian network with cloned anti-network.

Now our formulation is:

$$f_{\hat{L}\hat{M}}(J,K) = \underbrace{\frac{1}{Z}\frac{1}{f_{L,M}(J,K)}}_{\pi} = \sum_{\hat{L},\hat{M}} \underbrace{p(\hat{c}|\hat{L},\hat{M})}_{\gamma}\underbrace{p(\hat{L}|J)}_{\phi}\underbrace{p(\hat{M}|K)}_{\psi} \tag{6.3}$$

Which expands out in the following way, we express the formula as a set of functions, $g_{jk}(\mathbf{X})$, that are equal to zero.

$$\begin{aligned} \pi_{jk} &= \gamma_{11}\phi_j\psi_k + \gamma_{21}(1-\phi_j)\psi_k + \gamma_{12}\phi_j(1-\psi_k) + \gamma_{22}(1-\phi_j)(1-\psi_k) \\ 0 &= \gamma_{11}\phi_j\psi_k + \gamma_{21}(1-\phi_j)\psi_k + \gamma_{12}\phi_j(1-\psi_k) + \gamma_{22}(1-\phi_j)(1-\psi_k) - \pi_{jk} \end{aligned} \tag{6.4}$$

$$\begin{aligned} g_{jk}(\mathbf{X}) = \gamma_{11}\phi_j\psi_k + \gamma_{21}\psi_k - \gamma_{21}\phi_j\psi_k + \gamma_{12}\phi_j - \gamma_{12}\phi_j\psi_k + \gamma_{22} - \gamma_{22}\phi_j - \\ \gamma_{22}\psi_k + \gamma_{22}\phi_j\psi_k - \pi_{jk} \end{aligned} \tag{6.5}$$

There are four multivariate, polynomial functions over eight variables and we need to find $\mathbf{X}$ such that $g_{jk}(\mathbf{X}) = 0$ for all j and k.

Lower and Upper Bounds

Consider the situation where all of the γ variables equal zero. This is equivalent to a constraint that none of the parents of the *c-node* are true. Our conditioning to true in this case would be a zero probability event. This is only used for the purposes of having a lower bound for the $g_{jk}(\mathbf{X})$ function. In equation 6.4 we see that every term but the constant contains a γ variable and so the result will be all zeros leaving $g_{j,k}(\mathbf{X}) = -\pi_{j,k}$. The other variables, ϕ_j and ψ_k, have no affect in this case, so we can set them all to zero as well.

Since we are only interested in the point where $g_{jk}(\mathbf{X}) = 0$ there is no risk of attempting to condition on a zero probability outcome. The only situation when $g_{jk}(\mathbf{X}) = 0$ will be if $\pi_{jk} = 0$ as well. This is a degenerate case of the zero factor discussed in theorem 1.

Now consider the case where all variables are set to one. In equation 6.4 we see that all but two of the terms contain a $(1-\phi)$ or $(1-\psi)$ which are both zero under this assignment. We are left with $g_{jk}(\mathbf{X}) = 1 - \pi_{jk}$. Since $\pi_{jk} \in [0, 1]$ this shows that

$$g_{jk}(\mathbf{0}) \leq 0 \leq g_{jk}(\mathbf{1}).$$

We also know that polynomials are always continuous over the real numbers and so by the Intermediate Value Theorem each function is guaranteed to have a solution $x_{jk} \in \mathbf{X}$ such that

$$\begin{aligned} q_{11}(x_{11}) &= 0 \\ q_{12}(x_{12}) &= 0 \\ q_{21}(x_{21}) &= 0 \\ q_{22}(x_{22}) &= 0 \end{aligned}$$

Conjecture

In order for the solution to fully satisfy the requirements it must also be the case that $x_{11} = x_{12} = x_{21} = x_{22}$. That is, all four functions must cross zero at the same point. Unfortunately, we have not yet been able to prove that this is always the case. However, based on our experimental results we believe that this is so, and leave it as a conjecture. We have found solutions for some networks and we see no obvious reason why some network structures would lend themselves to solution by this method more than others.

6.4 General Formulation

Everything that was said for this specific example is also true in general. Figure 6.5 portrays the general case. The following formula represents the joint probability of the shield nodes and their ancestors, $\mathbf{A} = anc(\mathbf{S_C})$

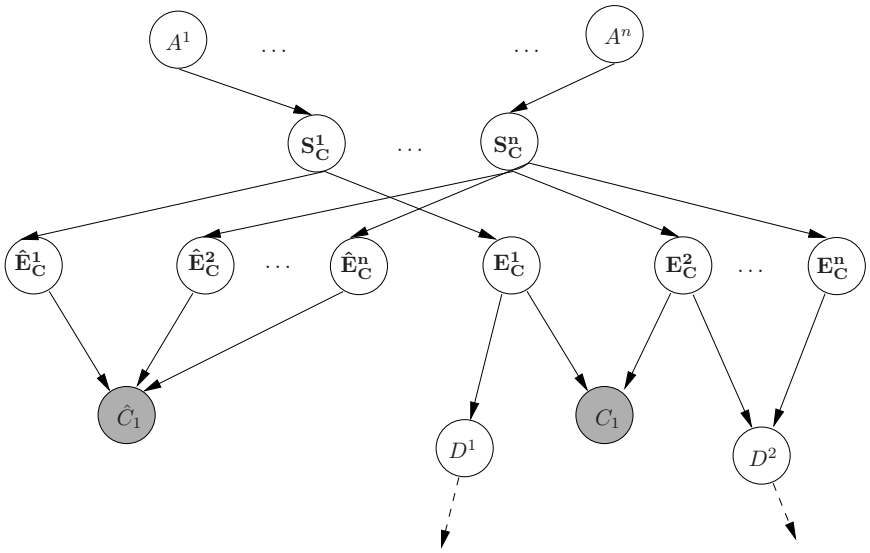


Figure 6.5: A general Bayesian network structure with anti-network.

$$
\begin{aligned}
p(\mathbf{A} \cup \mathbf{S_C}) = & \sum_{\hat{\mathbf{E}}^1_\mathbf{C} \ldots \hat{\mathbf{E}}^n_\mathbf{C}, \mathbf{E}^1_\mathbf{C} \ldots \mathbf{E}^n_\mathbf{C}} p(\mathbf{A}) p(\mathbf{S_C}|\mathbf{A}) p(\hat{\mathbf{c}}|\hat{\mathbf{E}}_\mathbf{C}) p(\hat{\mathbf{E}}^\mathbf{1}_\mathbf{C}|\mathbf{S_C}) \ldots p(\hat{\mathbf{E}}^\mathbf{n}_\mathbf{C}|\mathbf{S_C}) \times \\
& p(c|\mathbf{E_C}) p(\mathbf{E}^\mathbf{1}_\mathbf{C}|\mathbf{S_C}) \ldots p(\mathbf{E}^\mathbf{n}_\mathbf{C}|\mathbf{S_C}) \\
= & \, p(\mathbf{A}) p(\mathbf{S_C}|\mathbf{A}) \sum_{\hat{\mathbf{E}}^1_\mathbf{C} \ldots \hat{\mathbf{E}}^n_\mathbf{C}} p(\hat{\mathbf{c}}|\hat{\mathbf{E}}_\mathbf{C}) p(\hat{\mathbf{E}}^\mathbf{1}_\mathbf{C}|\mathbf{S_C}) \ldots p(\hat{\mathbf{E}}^\mathbf{n}_\mathbf{C}|\mathbf{S_C}) \times \\
& \sum_{\mathbf{E}^1_\mathbf{C} \ldots \mathbf{E}^n_\mathbf{C}} p(c|\mathbf{E_C}) p(\mathbf{E}^\mathbf{1}_\mathbf{C}|\mathbf{S_C}) \ldots p(\mathbf{E}^\mathbf{n}_\mathbf{C}|\mathbf{S_C}) \\
= & \, p(\mathbf{A}) p(\mathbf{S_C}|\mathbf{A}) f_{\hat{\mathbf{E}}_\mathbf{C}}(\mathbf{S_C}) f_{\mathbf{E_C}}(\mathbf{S_C})
\end{aligned}
$$

We thus require that

$$
f_{\hat{\mathbf{E}}_\mathbf{C}}(\mathbf{S_C}) = \frac{1}{\mathbf{Z}} \frac{1}{f_{\mathbf{E_C}}(\mathbf{S_C})} \qquad \text{where} \quad \mathbf{Z} = \sum_{\mathbf{S_C}} f_{\mathbf{E_C}}(\mathbf{S_C})
$$

When this is true the factor $f_{\hat{\mathbf{E}}_\mathbf{C}}(\mathbf{S_C})$ computed from the anti-network will fully cancel out the corresponding portion of the original network, thus eliminating the side effects caused by the conditioning of C.

As in the basic example we adopt a variable notation to represent the free parameters and characterize the nonlinear equations. This will help us understand the space of the problem and the feasibility of finding solutions.

We will represent particular instances of assignments of values to the affected clone nodes and shield nodes by:

$$
\mathbf{e} \in dom(\hat{\mathbf{E}}_\mathbf{C}) \qquad\qquad \mathbf{s} \in dom(\mathbf{S_C})
$$

The variables $\gamma_{\mathbf{e}}$ represent an instance of $p(\hat{c} = \mathbf{t}|\mathbf{e})$ under the current assignment of values to e-nodes given by $\mathbf{e}$. Similarly, $\phi^j_{\mathbf{e},\mathbf{s}}$ represents $p(\mathbf{e}^j|pa_{\mathbf{s}}(\hat{E}^j_C))$ for the j^{th} *e-node* where the relation $pa_{\mathbf{s}}(\hat{E}^1_C)$ indicates the value of the parents of node $\hat{E}^1_C$ within the current assignment $\mathbf{s}$. We will also represent the constant values of the factor for the anti-network under the current assignment of *s-nodes* with $\pi_{\mathbf{s}} = \frac{1}{\mathbf{Z}}\frac{1}{f_{\hat{\mathbf{E}}_\mathbf{C}}(\mathbf{s})}$. The set of all these variables is $\mathbf{X} = \{\gamma_{\mathbf{e}}, \phi^j_{\mathbf{e},\mathbf{s}}\} \quad \forall \mathbf{e} \in dom(\hat{\mathbf{E}}_\mathbf{C}), \mathbf{s} \in dom(\mathbf{S}_\mathbf{C}), j \in [1, |\hat{\mathbf{E}}_\mathbf{C}|]$

$$\frac{1}{\mathbf{Z}}\frac{1}{f_{\hat{\mathbf{E}}_\mathbf{C}}(\mathbf{s})} = f_{\hat{\mathbf{E}}_\mathbf{C}}(\mathbf{s})$$

$$\pi_{\mathbf{s}} = \sum_{\mathbf{e}\in dom(\hat{\mathbf{E}}_\mathbf{C})} p(\hat{c}|\mathbf{e})p(\mathbf{e}^1|pa_{\mathbf{s}}(\hat{E}^1_C)) \dots p(\mathbf{e}^n|pa_{\mathbf{s}}(\hat{E}^n_C))$$

$$0 = \sum_{\mathbf{e}\in dom(\hat{\mathbf{E}}_\mathbf{C})} p(\hat{c}|\mathbf{e}) \prod_{j=1}^{n} p(\mathbf{e}^j|pa_{\mathbf{s}}(\hat{E}^j_C)) - \pi_{\mathbf{s}} \tag{6.6}$$

$$g_{\mathbf{s}}(\mathbf{X}) = \sum_{\mathbf{e}\in dom(\hat{\mathbf{E}}_\mathbf{C})} \gamma_{\mathbf{e}} \prod_{j=1}^{n} (\phi^j_{\mathbf{e},\mathbf{s}})^{(\mathbf{e}^j=\mathbf{t})}(1 - \phi^j_{\mathbf{e},\mathbf{s}})^{(\mathbf{e}^j=\mathbf{f})} - \pi_{\mathbf{s}} \tag{6.7}$$

Since the sum ranges over *all* values of the *e-nodes*, not just true ones, we need the indicator power (*e.g.* $(\mathbf{e}^j = \mathbf{t})$), for the ϕ parameters in order to choose the appropriate form, negated or not, based on the value of $\mathbf{e}^j$. If $\mathbf{e}^j = \mathbf{t}$ then the exponent is 1. If $\mathbf{e}^j = \mathbf{f}$ then it equals 0.

We now have $|\mathbf{S}_\mathbf{C}|$ multivariate, polynomial functions given by $g_{\mathbf{s}}$ which we will call the *shield functions*. The goal is to find a solution $\mathbf{X}$ such that $g_{\mathbf{s}}(\mathbf{X}) = 0$ for all $\mathbf{s} \in dom(\mathbf{S}_\mathbf{C})$.

Size of Nonlinear System

Each equation contains the same γ variables, one in each term except for the final constant, $\pi_{\mathbf{s}}$, and there are $|dom(\hat{\mathbf{E}}_\mathbf{C})|$ terms. Each equation then contains a certain number of ϕ variables. It will be the same number in each equation but different depending on the values of the parents given the current assignment $\mathbf{s}$. In total, each *e-node* contributes $|dom(pa(\hat{E}^j_C))|$ variables to the entire system of equations, one for each unique assignment of its parents nodes. Thus the total number of variables is

$$n_v = |dom(\hat{\mathbf{E}}_\mathbf{C})| + \sum_{j=1}^{|\hat{\mathbf{E}}_\mathbf{C}|} |dom(pa(\hat{E}^j_C))| \tag{6.8}$$

The number of equations is simply the number of assignments of $\mathbf{s}$, which is $|dom(\mathbf{S}_\mathbf{C})|$. We cannot say anything more specific about the relationship between variables and equations. Depending on the domain sizes of nodes in $\hat{\mathbf{E}}_\mathbf{C}$ and $\mathbf{S}_\mathbf{C}$ and their connectivity there could be less, more or an equal number of variables as there are equations.

Lower Bound

Consider the point where all γ and all ϕ variables are set to zero. The the sum in equation (6.7) will clearly always result in zero thus leaving us with

$$g_{\mathbf{s}}(\mathbf{0}) = -\pi_{\mathbf{s}} \tag{6.9}$$

Since $\pi_{\mathbf{s}} \in [0, 1]$ for all $\mathbf{s} \in dom(\mathbf{S_C})$ this shows that

$$g_{\mathbf{s}}(\mathbf{0}) \leq 0 \quad \text{for all } \mathbf{s} \in dom(\mathbf{S_C}) \tag{6.10}$$

Upper Bound

Consider now the point when all γ and ϕ variables are set to one. In equation (6.7) we see that every term produced by the sum contains a $\phi^j_{\mathbf{e},\mathbf{s}}$ or a $(1 - \phi^j_{\mathbf{e},\mathbf{s}})$ term. The latter terms will all result in zero with this assignment. This leaves only one term produced by the instance $\mathbf{e}$ when all *e-nodes* are true. Since all the γ variables are also set to one, this term results in 1, giving us

$$g_{\mathbf{s}}(\mathbf{1}) = 1 - \pi_{\mathbf{s}} \tag{6.11}$$

And since $\pi_{\mathbf{s}} \in [0, 1]$

$$q_{\mathbf{s}}(\mathbf{1}) \geq 0 \tag{6.12}$$

We know that polynomials are always continuous over the real numbers and so by the Intermediate Value Theorem each function is guaranteed to have a solution $\mathbf{X'}$ such that $g_{\mathbf{s}}(\mathbf{X'}) = 0$ for all $\mathbf{s} \in \mathbf{S_C}$.

Conjecture

Unfortunately, we cannot yet prove the final step that there is some assignment of variables $\mathbf{X}^*$ such that $g_{\mathbf{s}}(\mathbf{X}^*) = 0$ for all $\mathbf{s} \in \mathbf{S_C}$, that is, that all of the $\mathbf{X'}$ are in fact, the same assignment.

We leave it as a conjecture that this is so, to be proven in future work. Note however, that if it is the case that certain classes of networks or distributions do not have solutions, anti-factors are still apply generally. Anti-networks would then still be a useful method of improving the complexity of a shielded network when the given model allows.

Chapter 7

Finding a Solution

The form that we have now arrived at is one with many nonlinear functions across many unknowns. We have explored several analytical possibilities for finding a solution based on the problem domain but so far have found none that apply to any sufficiently large set of networks. A search approach has yielded more useful results. We have opted to use optimization techniques to find a solution since they are well understood with easy to access algorithms for anyone wanting to implement shielding. They also allow the easy specification of constraints which will be useful for our problem. We first provide a brief overview of the theory of constrained optimization and how it is used in the MATLAB optimization toolbox software package, which is the implementation that we use.

7.1 Constrained Optimization Background

The topic of constrained optimization is a wide-ranging and complex one. Here we give only the briefest overview to indicate what is necessary for our problem. If you wish to find out more a very clear description of these techniques and many others is provided in [15].

A constrained optimization problem, also called a nonlinear programming problem, is of the following form

$$\min_{\mathbf{x}\in\Re^n} G(\mathbf{x}) \qquad \text{such that} \qquad \begin{cases} c_i(\mathbf{x}) = 0 \\ c_j(\mathbf{x}) \geq 0 \end{cases} \qquad \text{are satisfied.} \qquad (7.1)$$

Where c is a vector of constraint functions to be satisfied and $i \in [1, m], j \in [m+1, n]$ where n is the number of constraints.

7.1.1 The KKT Conditions

Many algorithms for solving nonlinear programming problems centre around the *Karush-Kuhn-Tucker conditions* [10]. They state that given linear independence among the active constraint gradients at a local solution $\mathbf{x}^*$, the following

conditions, in addition to 7.1, are necessary for optimality of $\mathbf{x}^*$:

$$\nabla G(\mathbf{x}^*) - \sum_{k=1}^{n} \lambda_k^* \nabla c_k(\mathbf{x}^*) = 0, \tag{7.2}$$

$$\lambda_j^* \geq 0, \qquad \text{for all } j \in [1, m], \tag{7.3}$$

$$\lambda_k^* c_k(\mathbf{x}^*) = 0 \qquad \text{for all } k \in [1, n]. \tag{7.4}$$

When a solution is found, the KKT conditions tell us that the gradients of the objective function and constraints will cancel each other out given some set of lagrange multipliers given by λ. Note that this only holds for *active* constraints, that is, when $c_k(\mathbf{x}^*) = 0$ for all $k \in [1, n]$. Thus any inequality constraints where $c_j(\mathbf{x}) > 0$ will be left out by having $\lambda_j = 0$.

7.1.2 Sequential Quadratic Programming

This is a popular method used to solve the constrained optimization problem, and it is the method used by MATLAB through its `fmincon` function. The method works by solving a quadratic programming subproblem multiple times. At each step it determines the search direction as a solution to a quadratic subproblem of (7.2) with linearized constraints. The goal is to compute a new estimate of the lagrange multipliers and a search direction to improve the estimate of $\mathbf{x}^*$ until some merit function is satisfied. These new estimates are then fed back into the next iteration of the algorithm. These methods have guaranteed convergence rates that are better than linear and often perform very well in practice.

For a comprehensive and very clear explanation of these method see [15]. For other references on these techniques see the work done by Han [6] and Powell [21]. A literature review is provided in [2].

7.2 Reformulation of Problem

Using the shielding functions, $g_{\mathbf{s}}(\mathbf{X})$ from equation 6.7 we reformulate our problem to be solved as a constrained nonlinear system. Starting from a point where $g_{\mathbf{s}}(\mathbf{X}) > 0$, we want to minimize an objective function, $G(\mathbf{X})$, that is a linear combination of the $g_{\mathbf{s}}$ functions subject to the constraint that each of the components of G remains non-negative at all times. Being a linear combination will ensure that if $g_{\mathbf{s}}(\mathbf{X}) = 0$ then $G(\mathbf{X}) = \mathbf{0}$. Non-negativity allows us to avoid incorrectly identifying as an answer the situation when $G(\mathbf{X}) = 0$ but $g_{\mathbf{s}}(\mathbf{X}) \neq 0$ due to some of the functions being positive and some negative in a such a way that they exactly balance out.

We define the objective function as

$$G(\mathbf{X}) = \sum_{\mathbf{s} \in dom\mathbf{S})} g_{\mathbf{s}}(\mathbf{X}) \tag{7.5}$$

We have natural constraints in that the shielding functions that are the components of $G(\mathbf{X})$ are always zero. However, we do not use c_i to specify this strong equality constraint, since this would require specifying a starting point that already satisfies the constraints. If we have such a starting point then this would be an acceptable final answer for our problem. So in our formulation of (7.1), we have $m = 0$ and $n = |dom(\mathbf{S_C})|$. We define an ordering over the values of the *s-nodes* $\mathbf{s} \in dom(\mathbf{S_C})$ indexed by $j \in [1, n]$ so that $g_j(\mathbf{X})$ is the shielding function corresponding to the j^{th} value of $\mathbf{s}$. We specify a set of constraints that require the shielding functions to always be non-negative :

$$c_j(\mathbf{X}) = -g_j(\mathbf{X}) \tag{7.6}$$

The functions must be negated in order to reverse the inequality in (7.1). The starting point here need only satisfy $g_{\mathbf{s}}(\mathbf{X}) \geq 0$.

Additionally, though not a constraint on the function, we must restrict the domain of the variables to the range $[0, 1]$ since each variable must represent a probability. We cannot simply drop this restriction and normalize the variables X in order to retrieve probabilities since the shielding functions contain products and division will not distribute over them.

7.3 Example Networks

Our solutions were computed using the MATLAB Optimization Toolbox implementation [8] of the SQP algorithm through its `fmincon` function. This required writing MATLAB function forms of the functions $G(\mathbf{X})$ and $c_j(\mathbf{X})$ for all $j \in [1, n]$. We also made extensive use of Kevin Murphy's *BNT Toolbox* [14] for MATLAB which allows representation and computation of Bayesian networks within that environment.

Instructor Example

First we return to the instructors in example 4. After converting the network to the general formulation and expressing it as a nonlinear constrained optimization problem, we find the values for $\mathbf{X}$, shown in figure 7.1. Figure 7.2 shows the computed beliefs after inference, which conforms to our desired distribution.

Note that these are the same distributions determined by both the anti-factor method in 5.5 and the clique method in 4.3. Comparing just to the anti-factor approach we also see in figures 7.3-7.6 that further observations propagate and obey the original constraint in the same way.

Localized Effect

An important feature of networks containing conditioning and shielding is that the counterbalancing of the two is fully localized. This means that adding additional ancestors or descendants outside the sets $\mathbf{S_C}$ and $\mathbf{E_C}$ can be done arbitrarily without the needs for recomputing the anti-network. Some examples of this are shown in figures 7.7 and 7.8.

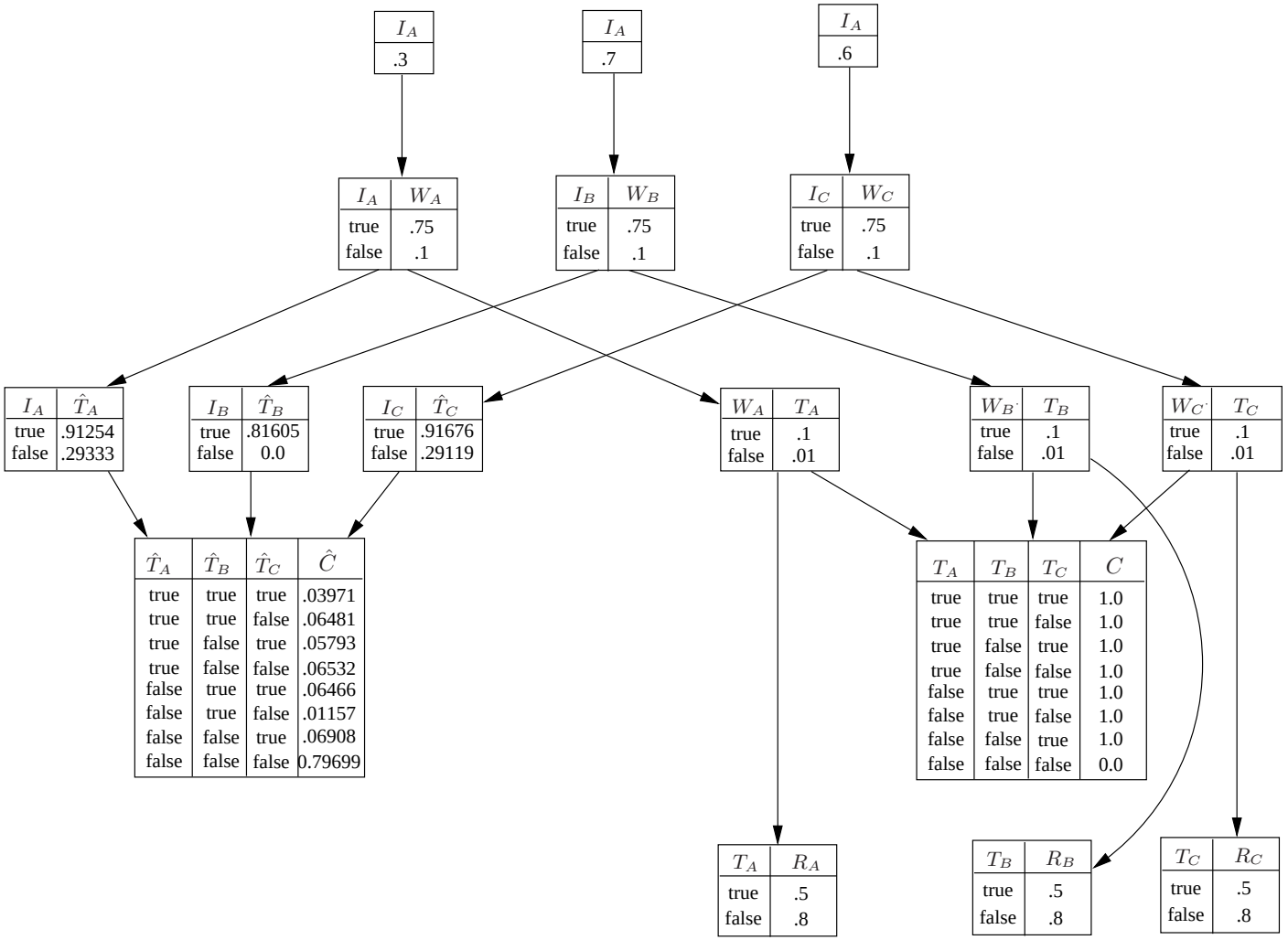


Figure 7.1: Computed CPDs for anti-network found using nonlinear constrained optimization.

Multiple *c-nodes* across network

Figure 7.9 shows a Bayesian network with two *c-nodes*, C_1 and C_2, and augmented with two cloned anti-networks. Each set of cloned nodes compensates for a different *c-node*. The CPDs for $\hat{L}, \hat{M}, \hat{C}_2$ were solved separately from those for $\hat{T_A}, \hat{T_B}, \hat{T_C}, \hat{C}_1$. They can in fact be solved as separate Bayesian networks and connected together afterwards, as long as no *e-nodes* are shared in common as parents of multiple *c-nodes*.

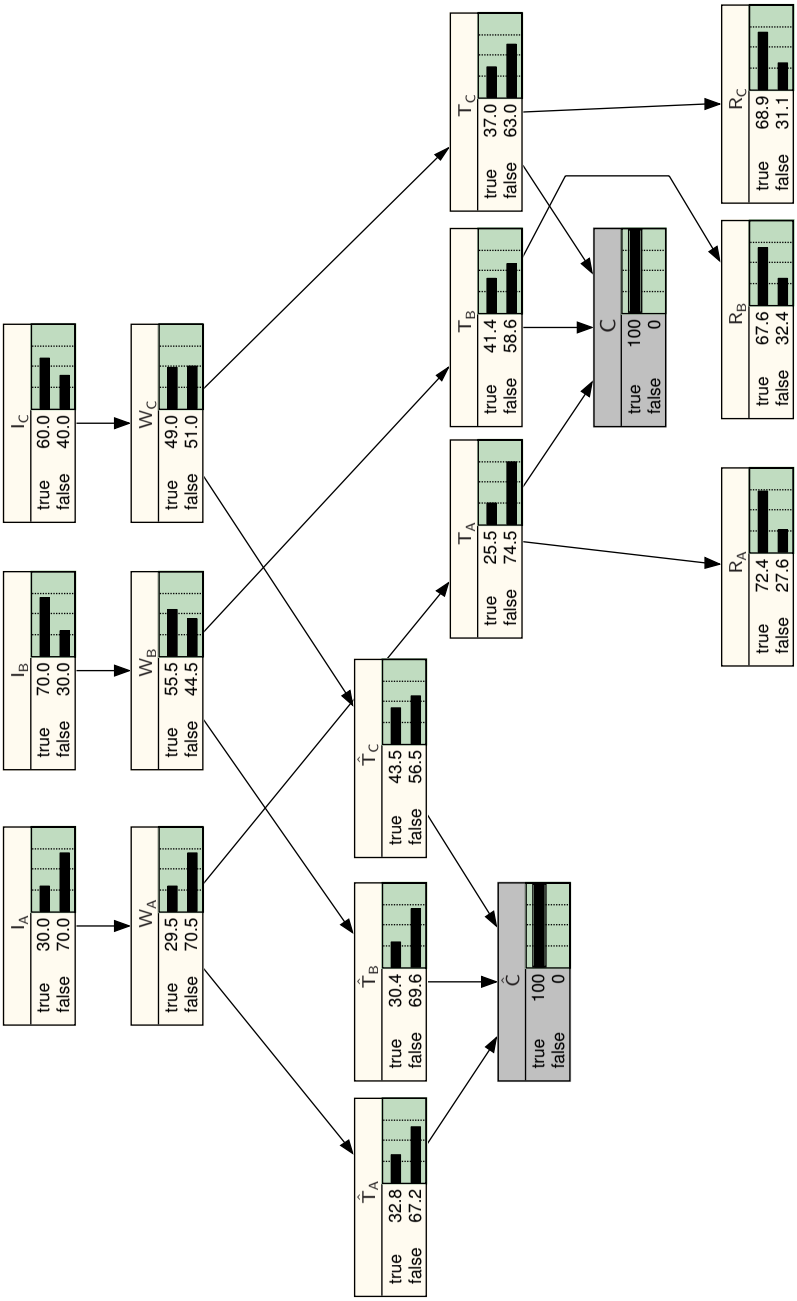


Figure 7.2: Computed beliefs after inference for a shielded network using a cloned anti-network for example 4.

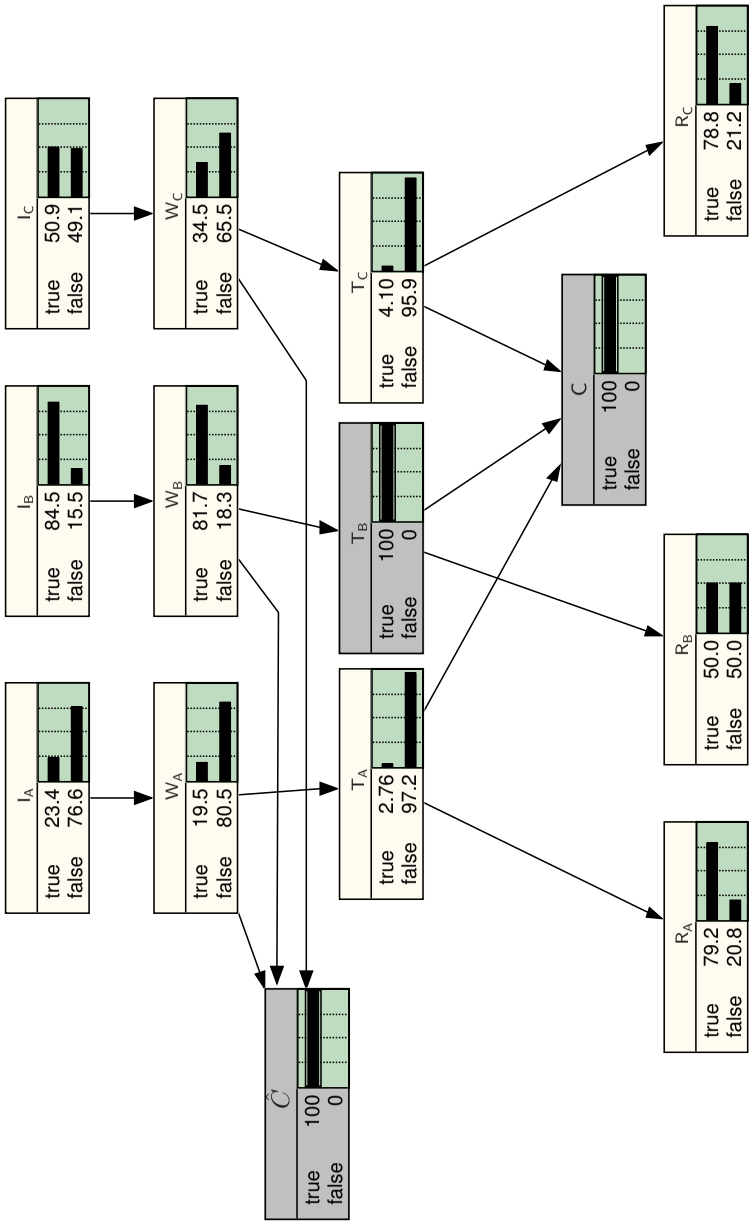


Figure 7.3: Distribution after conditioning $T_B = true$ using an anti-factor.

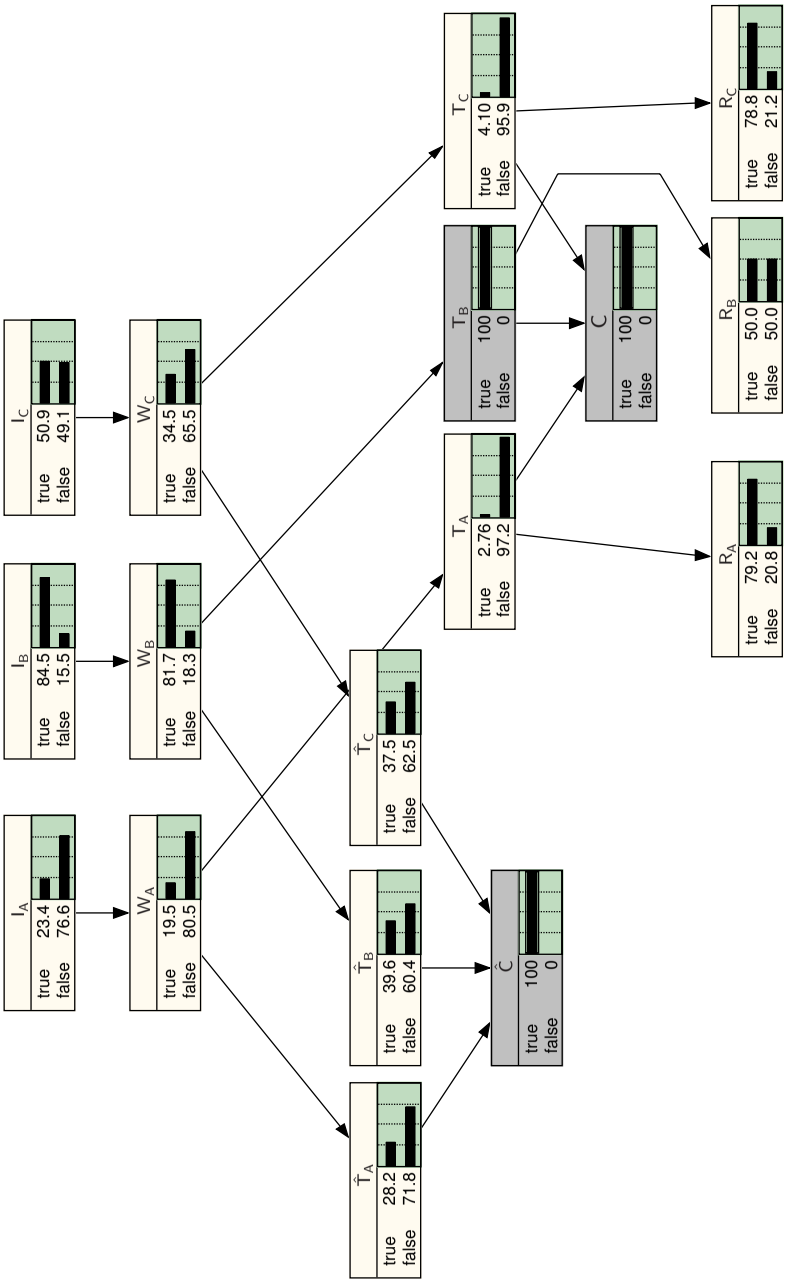


Figure 7.4: Distribution after conditioning $T_B = true$ using an anti-network.

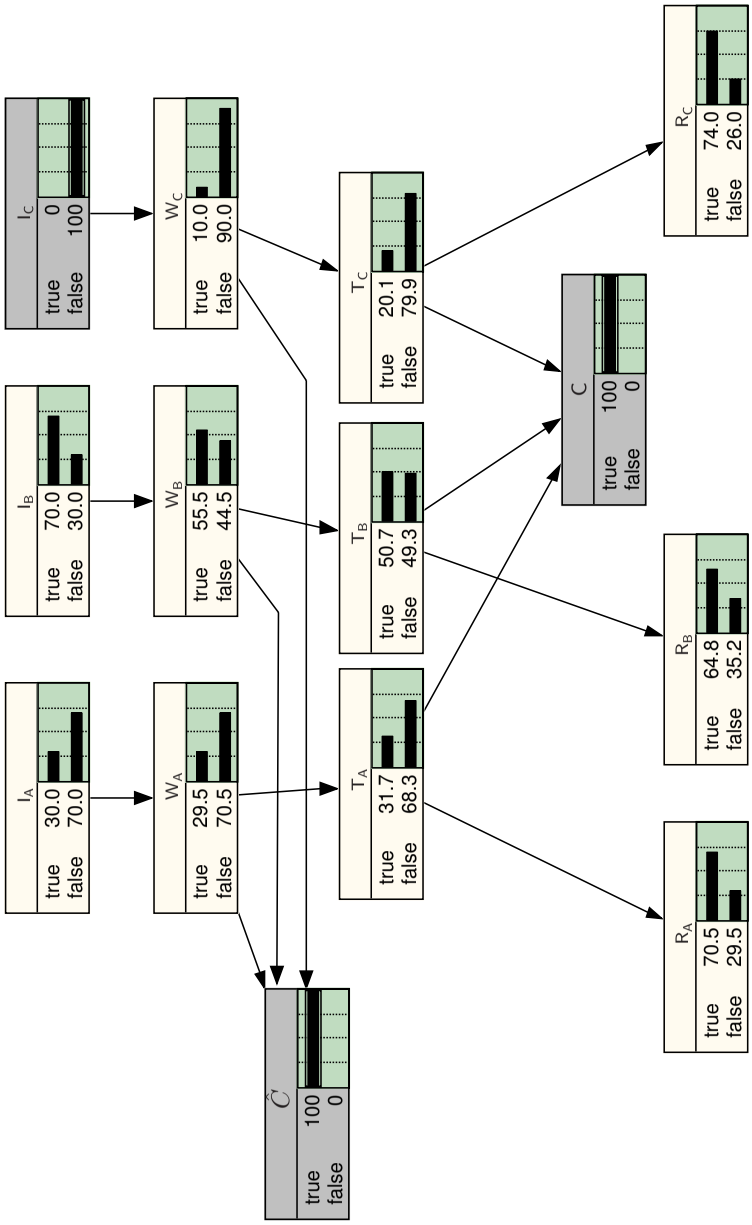


Figure 7.5: Distribution after observing $I_C = false$ using an anti-factor.

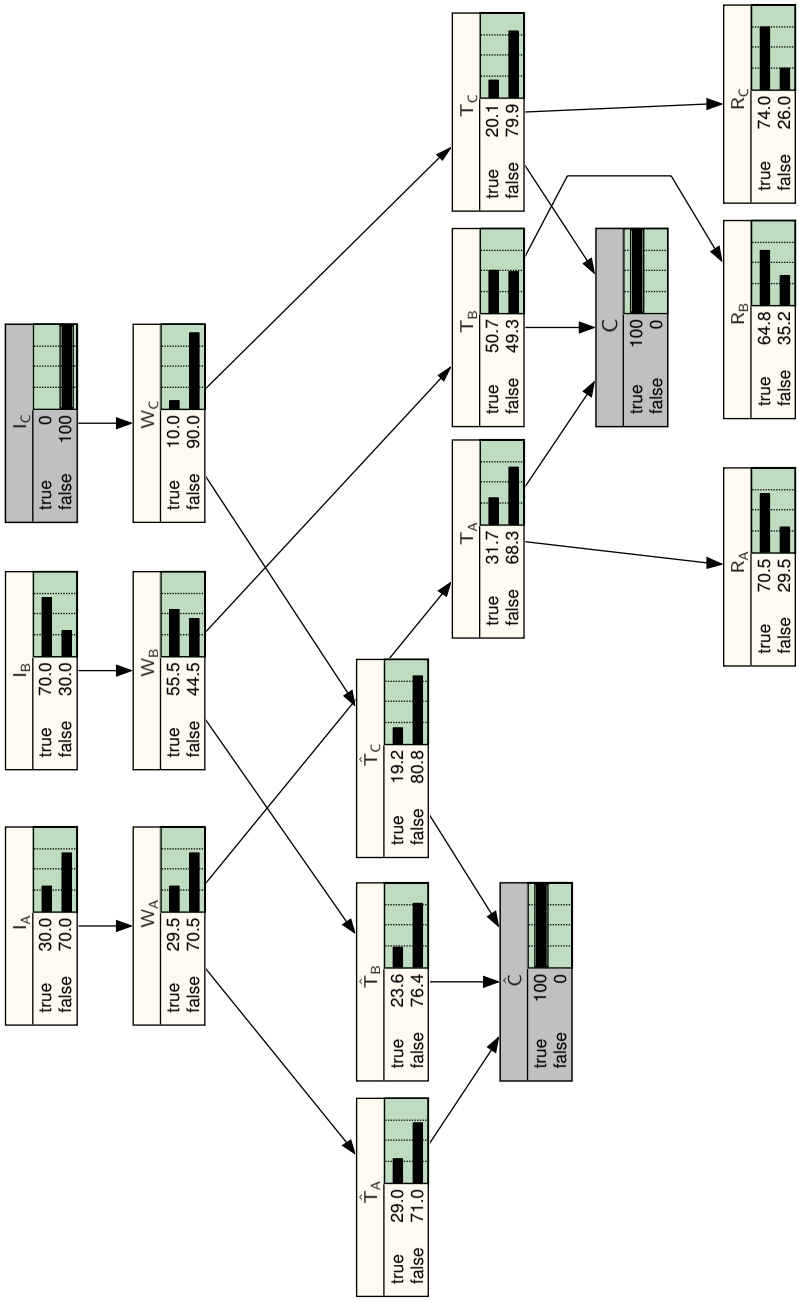


Figure 7.6: Distribution after observing $I_C = false$ using an anti-network.

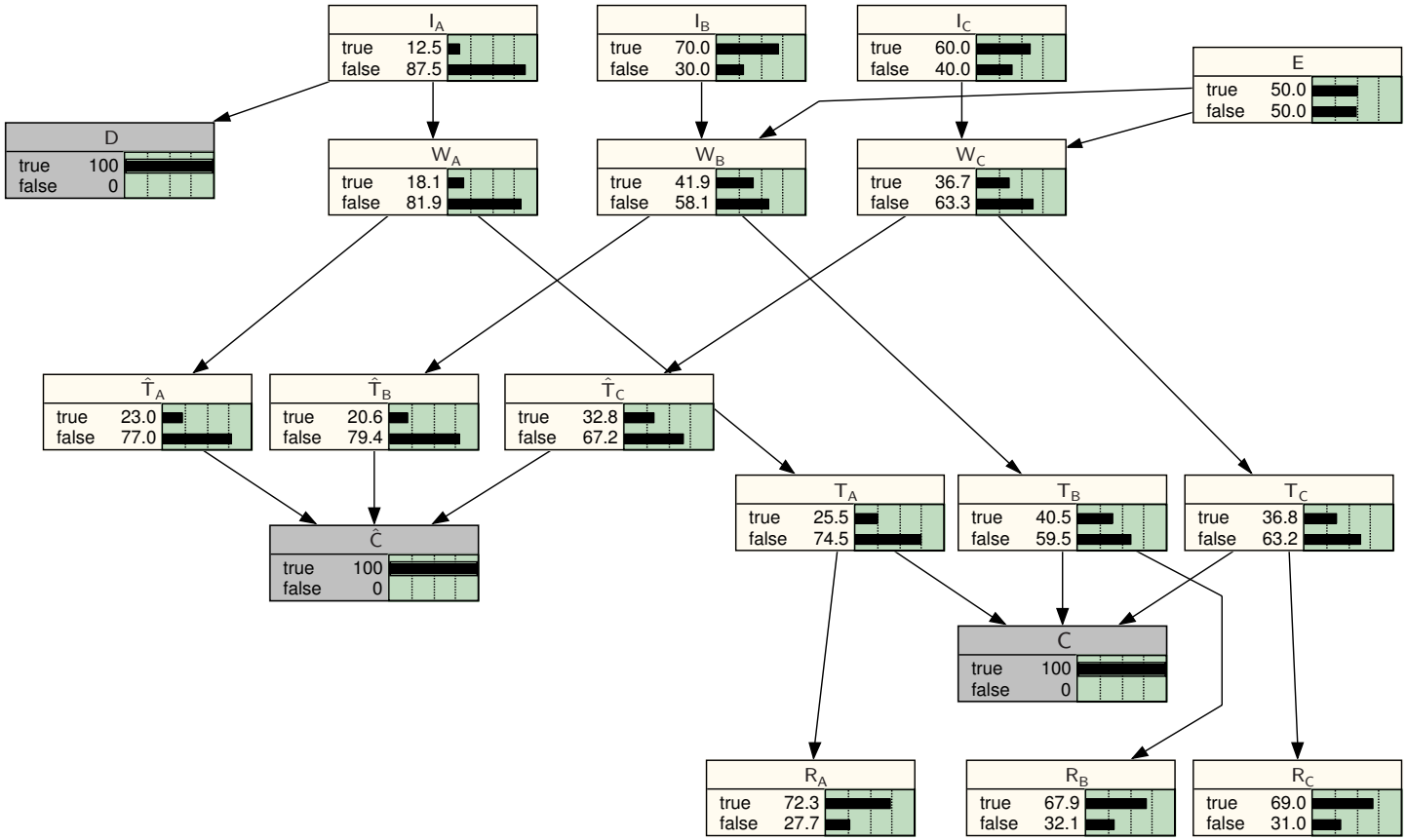


Figure 7.7: Observing $D = true$ reduces our belief that Alice is interested in AI, this influences the distribution of the T_X nodes but Bob and Cindy's interests remain shielded. E also maitains its original distribution. The antinetwork does not need to be recomputed to add D.

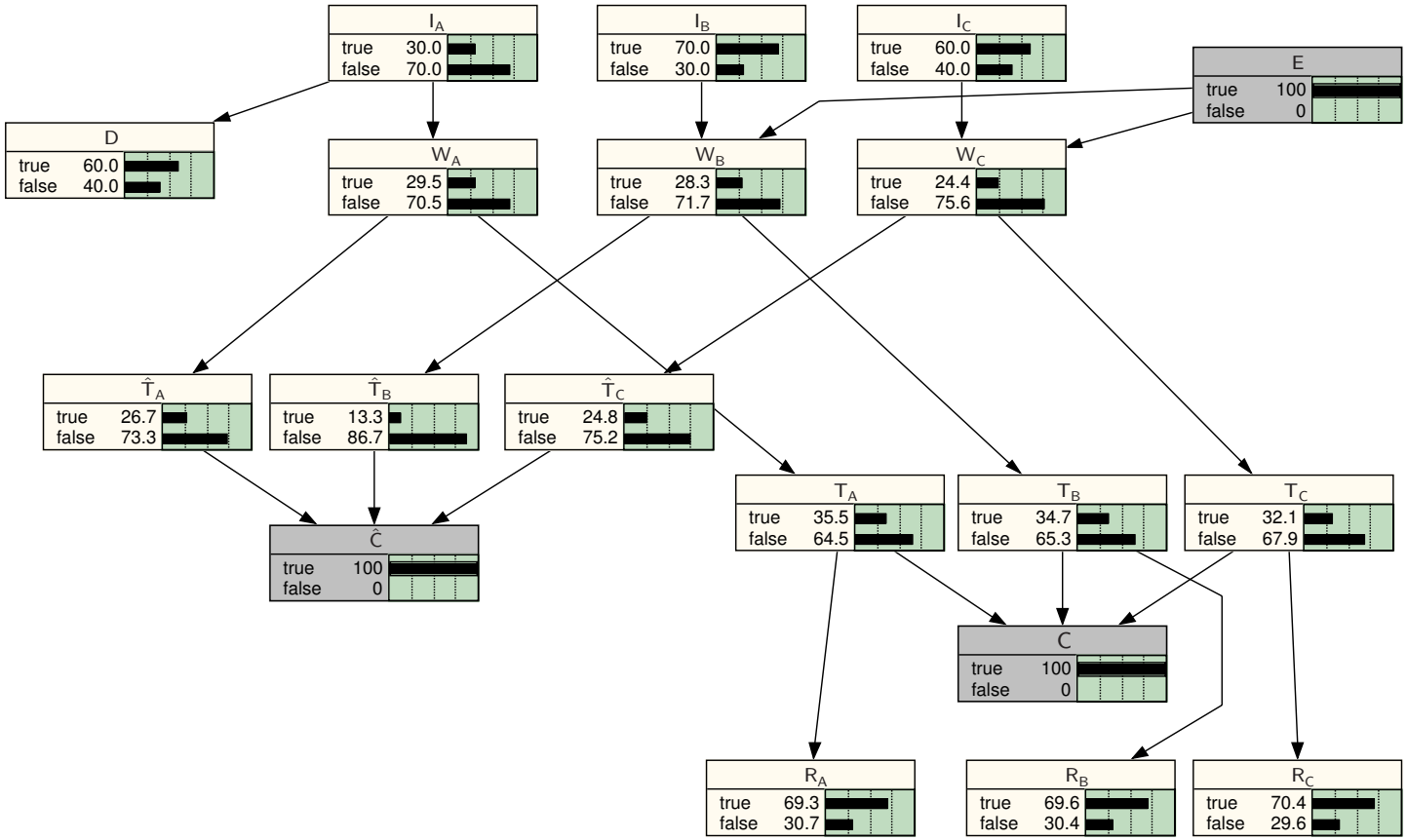


Figure 7.8: Observing $E = true$ reduces our belief in the desire to teach for Bob and Cindy, this influences the distribution of the T_X nodes but Alice's interest is unaffected. D also maintains its original distribution. The antinetwork does not need to be recomputed to add E.

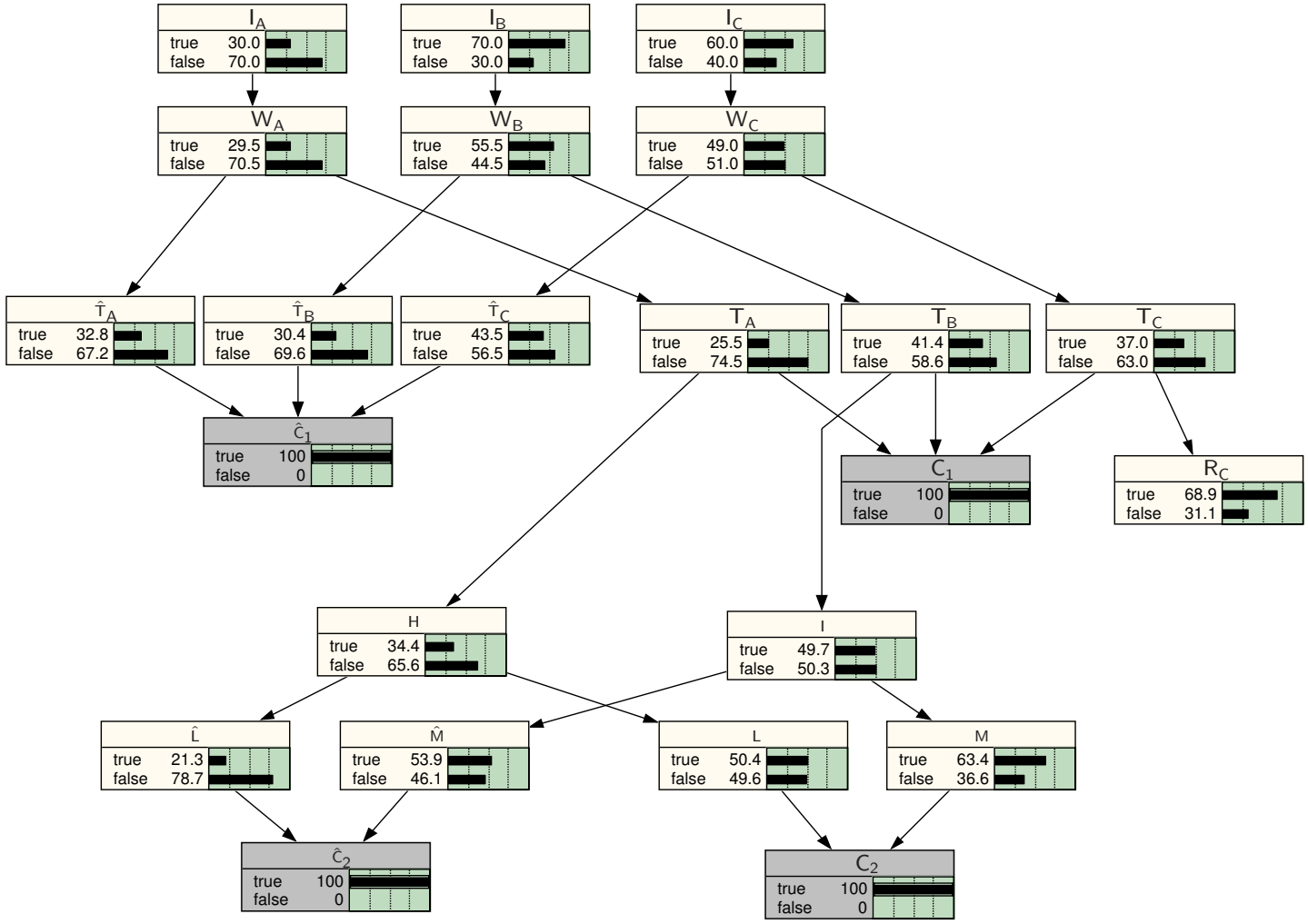


Figure 7.9: Bayesian network containing two *c-nodes*, each shielded seperately. Notice that the beliefs for T_X, W_X and I_X nodes are the same as before.

Chapter 8

Conclusion

The motivation of this work has been the specification of distributions in directed graphical models that are broken into two distinct components known to the modeler. For some subset of variables in the model there are known directed, or conditional, distributions, and an undirected, or constrained, distribution. The modeler presented with representing such a distribution in a Bayesian network finds that significant calculations must first be made and the resulting network is often an unintuitive or forced representation of the problem. The use of an added node conditioned to true to represent the undirected component of the distribution is much more straightforward for those familiar with the behaviour of evidence in Bayesian networks. It also requires no calculations on the part of the modeler to specify the distribution.

Unfortunately, this technique does not always lead to the desired distribution, causing dependencies among ancestor nodes not implied by the original problem. We have introduced this problem of side effects of conditioning and presented a general method for counteracting them with the use of computed anti-factors. When the network structure leads this method to large complexity increases we have shown that a conditional structure can be created, a copy of the original affected network, called a cloned anti-network. The parameters of the distribution for this anti-network need to be computed. We used nonlinear constrained optimization for this with very good results. The resulting networks provide the modeler with the best of both worlds, a straightforward specification matching the knowledge they have available and a lack of side effects caused by the use of conditioning to obtain the distribution.

8.1 Open Problems

The methods introduced here could be used as part of a Bayesian network modelling system to provide the modeler with a post-specification processor to create the shielded anti-network. This would have the advantage of allowing them to see both distributions to determine if shielding is necessary for their situation as the distinction is sometimes subtle.

Some open problems that involve a more general bound on the size of the cliques that are added to the junction tree when a model is augmented with an anti-network. The promise of the anti-network technique is partially in the simplicity and symmetry of its structure. It remains to be proven fully however that a solution to the anti-network parameters always exists, in practice we have

always found results. Showing this existence or showing some set of networks where a solution must exist may help further in refining algorithms for searching the parameter space. For distributions where the constraint component is a function which can be represented compactly, such as "or" it is unknown if any simplifications can be made to the resulting distribution of nodes in the anti-network.

There are many connections between the use of conditioning to induce undirected distributions and undirected graphical models such as Markov Random Fields. Conditioning can be used to simulate undirected distributions within directed models. The idea of side effects has not been studied in undirected models as of yet but the results of such study would no doubt be enlightening for an understanding of both frameworks and the connection between the them. The shielded distributions created in Bayesian networks using anti-networks and conditioned nodes, for example, are difficult to model in MRFs. There are also interesting questions relating to conditioning in other directed modelling languages such as Influence Diagrams such as the ramifications of side effects for utility maximization and the meaning of a conditioned node constraining decisions.

Bibliography

[1] Valeria Bertacco and Maurizio Damiani. The disjunctive decomposition of logic functions. In *ICCAD '97: Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 78–82, Washington, DC, USA, 1997. IEEE Computer Society.

[2] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 1996.

[3] Wray L. Buntine. Chain graphs for learning. In *Uncertainty in Artificial Intelligence*, pages 46–54, 1995.

[4] Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

[5] Rina Dechter. Bucket elimination : A unifying framework for probabilistic inference. In E. Horvitz and F. Jensen, editors, *Proceeding of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 211–219, 1996.

[6] S. P. Han. Superlinearly convergent variable metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11:263–282, 1976.

[7] Norsys Software Inc. Netica : Bayesian network modelling tool. http://www.norsys.com/netica.html, October 2005.

[8] The MathWorks Inc. Matlab optimization toolbox, October, 2005. http://www.mathworks.com/access/helpdesk/help/toolbox/optim/.

[9] Finn V. Jensen. Junction trees and decomposable hypergraphs. Technical report, Judex Datasystemer, Aalborg, Denmark., 1988.

[10] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceeding of teh Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, 1951.

[11] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. In *Readings in uncertain reasoning*, pages 415–448. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[12] Alan Mackworth. *Encyclopdeia of Artificial Intelligence*, chapter Constraint Satisfaction, pages 285–293. John Wiley and Sons, 1992.

[13] K. Murphy. Inference and learning in hybrid bayesian networks. Technical Report Technical Report 990, University of California, Berkeley, 1998.

[14] Kevin Murphy. Bayes net toolbox for matlab. http://www.cs.ubc.ca/ murphyk/Software/BNT/bnt.html, October, 2005.

[15] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag New York, Inc., 1999.

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

[17] Judea Pearl and Tom S. Verma. A theory of inferred causation. In James F. Allen, Richard Fikes, and Erik Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 441–452, San Mateo, California, 1991. Morgan Kaufmann.

[18] David Poole, Alan Mackworth, and Randy Goebel. *Computational Intelligence : A Logical Approach*. Oxford University Press, 1998.

[19] Shachter Ross. Bayes-ball: The rational pasttime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 480–487, San Francisco, CA, 1998. Morgan Kaufmann Publishers.

[20] Stuart Russell and Peter Norvig. *Artificial Intelligence : A Modern Approach*. Prentice Hall, 1995.

[21] G. A. Watson, editor. *A fast algorithm for nonlinearly constrained optimization calculations*. Springer Verlag, Berlin, 1977.

[22] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium*, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[23] Nevin Zhang and David Poole. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.