IB Subject(s): Mathematics

Title – Inference in Bayesian networks

Research question: Is variable elimination an efficient method of inference in Bayesian networks, and how can changing the ordering in which variables are eliminated affect the efficiency of variable elimination?

(Word Count: 3946)

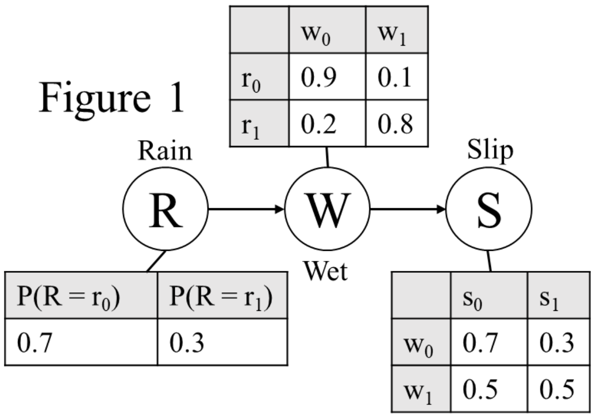
Contents

Introduction

The ‘Butterfly theory’ is the idea that a butterfly flapping its wings on one side of the earth can cause a storm on the other side. Upon learning about this theory, I wondered whether the world would have been completely different had I not been born. After all, the theory says that the most minor of events can eventually lead to catastrophic ones. This theory is based upon ‘Causality’. Causality is the idea that events influence other events to occur. For instance, on a certain day, it rains. This results in the ground being wet. If the ground is wet, there is a higher chance one may slip and fall.

Bayesian networks are a graphical representation of causality and conditional probability, which, in addition to showing the events, show the likelihoods of events. They are Directed Acyclic Graphs (DAGs) where the nodes represent a random variable, and the edges represent the flow of influence. Each node has a Conditional Probability Table (CPT) linked to it, which represents the probability distributions for that node given its parents (the nodes that influence it). Figure 1 is a Bayesian network representing the previously mentioned example. The numbers assigned here are arbitrary. However, they follow the logic that if it rains, then the chance of the ground becoming wet is higher than if it doesn’t rain, and that if the ground is wet, then the chance of slipping is higher than if the ground is not wet.

Figure 1



	w_0	w_1
r_0	0.9	0.1
r_1	0.2	0.8

$P(R = r_0)$	$P(R = r_1)$
0.7	0.3

	s_0	s_1
w_0	0.7	0.3
w_1	0.5	0.5

The Chain Rule for Bayesian Networks states that a certain probability over a set of random variables can be calculated using only conditional probabilities. The Chain rule can be applied to Figure 1 as follows:

$$P(R, W, S) = P(R) \times P(W|R) \times P(S|W)$$

This makes intuitive sense as the value of a node's grandparents does not influence the node if the node's parent is conditioned upon.

More generally,

$$P(A_1, \dots, A_n) = \prod_{i=1}^{n} P(A_i | Par(A_i))$$

$Par(A_i)$ refers to the parents of *A*$_i$. *n* is the number of nodes in the network.

Given a set of random variables, a joint probability distribution is a list of the probabilities of all permutations of values of the random variables. It can be formed by multiplying all the CPTs in a Bayesian network. A joint distribution table consisting of *n* random variables, each having *k* distinct values will have a length of $k^n - 1$. The last entry into the table does not need to be stored as it is complementary to *1* to the other entries. This means the length is exponential in relation to the number of random variables. Additionally, joint distribution tables do not intuitively show how random variables influence others. Representing the graph in Figure 1 as a Joint distribution would require storing *7* values, as the factor for *R* has *2* values, *W* has *2* values and *S* has *2* values.

Every CPT belongs to a 'factor'. A factor is a function that takes in random variables and returns a probability from its CPT. The scope of a factor is the random variables it takes as input. In

Figure 1, the factor representing W would be $P(W|R)$ and would have a scope of W and R. This factor can also be written as $\phi_W(W,R)$. The factor of S would be $P(S|W)$. It would have a scope of S and W. $P(S|W)$ can also be written as $\phi_S(S,W)$.

There are three operations that can be done on factors:

Conditioning: eliminating all other values of a random variable other than the one conditioned on and renormalizing the CPT. Renormalizing involves summing up all probabilities in the CPT and dividing each probability by that sum.

Marginalization: removing a random variable from the scope of a factor by summing the probabilities where the values of the other random variables are the same. The equation below represents the marginalizing of the variable N out of the arbitrary factor $\phi_1(M,N,O,P)$.

$$\phi_2(M,O,P) = \sum_N \phi_1(M,N,O,P)$$

Multiplication: ‘combining’ two factors by multiplying the rows which assign the same values to variables in common. This creates a factor with the combined scope of the two factors that were multiplied.

A conditional probability query consists of evidence $E = e$ and a set of query variables y. The task is to compute $P(Y|E = e)$. This is known as Inference [Huang].

Does tracing work?

When I learned about inference, I assumed that one could trace the path of influence from one node in a graph to another. In other words, I assumed that there were a limited number of paths as the flow of influence is only in one direction. However, I soon recognized that this is only considering causal reasoning, i.e. the reasoning which flows in the same direction as the edges [Koller, “Flow of Probabilistic Influence”].

There are two other ways to describe influence in a graph.

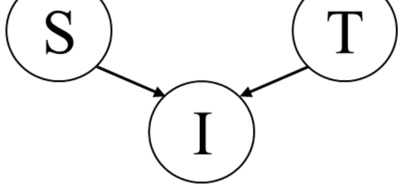
The first is evidential-based reasoning. This type of reasoning flows in the opposite direction of the edges. It can best be summarized by this statement: ‘Given observations of an event, what is the probability of the event that influences it occurring?’. This is analogous to the Bayes theorem. In Figure 1, rain influences the ground to be wet based on causal reasoning. $P(W|R)$ is *0.8*. However, if the ground is observed to be wet, i.e. conditioning on the ground being wet, the probability that it rained is influenced. This can be shown using the Bayes theorem:

$$P(R|W) = \frac{P(W|R)P(R)}{P(W|R)P(R) + P(W|R')P(R')} = \frac{0.24}{0.24 + 0.03} = 0.889$$

Now, by conditioning on wet ground, the probability of it to have rained is ~0.889 as compared to 0.7 when not conditioning.

The second is intercausal reasoning. If two nodes share a common child (i.e. have edges pointing at the same node), and the common child is conditioned upon, conditioning on one parent influences the other. Since this seemed counter-intuitive at first, I constructed an analogy which is represented by the graph below:

Figure 2



S represents the presence of salt and is binary in its values. Salt is present or not (*s*: salt is present, *s*': no salt). *T* represents temperature and is also binary. *T* can either be hot or cold (*h'*: cold, *h*: hot). *I* refers to the state of a piece of ice and is binary. It can either be ice or melted into water (*i*: not melted, *i'*: melted).

According to intercausal reasoning, when *T* is conditioned upon, *S* should be influenced and vice versa given that *I* is also conditioned upon. To intuitively understand this, I reasoned that if the ice was melted, and the temperature was cold, then the probability that salt was present must have been high. Likewise, if the ice was melted, and the temperature was warm, then the probability that salt was present must have been low.

More concretely, let the probability for each value of *S* and *T* be even i.e. 0.5 and 0.5, and let *I* follow the following values:

- $P(i|s,h) = 0.1$, $P(i'|s,h) = 0.9$
- $P(i|s',h') = 0.8$, $P(i'|s',h') = 0.2$
- $P(i|s,h') = 0.6$, $P(i'|s,h') = 0.4$
- $P(i|s',h) = 0.4$, $P(i'|s',h) = 0.6$

Note that these values are arbitrarily picked, however, are make sense in the analogy.

What would be the probability of salt being present if the temperature was cold, and the ice was melted? More formally, what is $P(S = s|I = \ i', H = h')$?

$$P(s|i',h') = \frac{P(i',s,h')}{P(i',h')} = \frac{P(i'|s,h')P(s)P(h')}{P(i'|s,h')P(s)P(h') + P(i'|s',h')P(s')P(h')}$$

$$= \frac{0.4 \times 0.5 \times 0.5}{0.4 \times 0.5 \times 0.5 + 0.2 \times 0.5 \times 0.5} = \frac{2}{3}$$

Likewise, what would be the probability of salt being present if the temperature was hot, and the ice was melted? Mathematically, what is $(S = s|I = i', H = h)$?

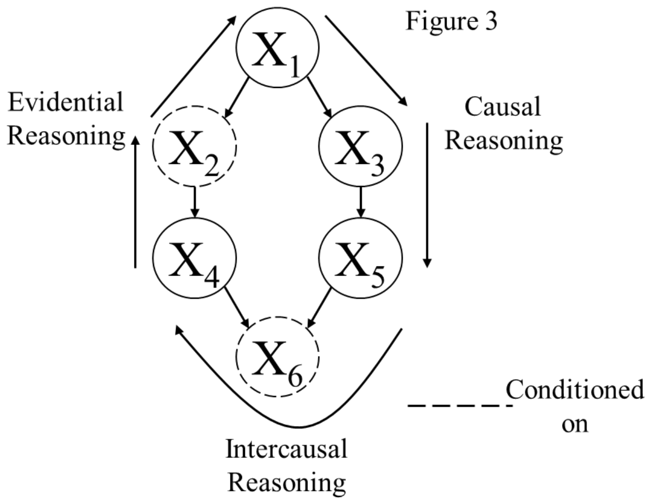
$$P(s|i',h) = \frac{P(i',s,h)}{P(i',h)} = \frac{P(i'|s,h)P(s)P(h)}{P(i'|s,h)P(s)P(h) + P(i'|s',h)P(s')P(h)}$$

$$= \frac{0.9 \times 0.5 \times 0.5}{0.9 \times 0.5 \times 0.5 + 0.6 \times 0.5 \times 0.5} = \frac{3}{5}$$

$\frac{3}{5} < \frac{2}{3}$. This makes intuitive sense. If the temperature was cold and the ice was melted, the chance of salt being present would have to be higher.

Considering these three ways of influence in a graph, one can see how there can be a high number of paths of influence between two nodes, especially in more complex graphs. Therefore, while tracing is possible, it is not an efficient form of inference.

Learning the ways in which influence flowed, I wondered why any graph isn't a loop. For instance, why isn't the figure below a loop?



I initially reasoned that if X_2 and X_6 are conditioned upon, then, by evidential reasoning, X_1 would be influenced. Following this, X_3 and hence X_5 would be influenced by causal reasoning.

Finally, by intercausal reasoning, X_4 would be influenced. X_4 would influence X_2 through evidential reasoning, and the loop repeats itself forever. Also, by a similar logic, when X_2 influences X_1 through evidential reasoning, why wouldn't X_1 influence X_2 in turn through causal reasoning?

I soon recognized that this logic is flawed. Since X_2 is already conditioned upon, I realized that it can't be influenced. Hence, it acts as a barrier that blocks influence from flowing in a loop. Hence, X_1 also cannot influence X_2 through causal reasoning.

Variable Elimination

There are two types of Inference:

- Exact Inference: given a conditional probability query, an exact value of a probability must be returned
- Approximate Inference: given a conditional probability query, a probability within a range of *r* of the actual probability must be returned.

Variable Elimination is a method of Exact inference [Malone]. It works on the premise of marginalization. To summarize, factors sharing a random variable are multiplied and then have the shared random variable marginalized. The marginalization is repeated across all random variables except the one that is queried. This will leave a factor that consists of probabilities for the queried random variable.

Let W be the set of all random variables that do not belong to E or Y. All values of W must be marginalized when calculating $P(Y|E)$.

Given Evidence set E, and a set of query variables y, two values must be calculated to find $P(Y|E)$: $P(Y, E)$ and $P(E)$.

This is because

$$P(Y,E) = P(Y|E) \times P(E)$$

$$P(Y|E) = \frac{P(Y,E)}{P(E)}$$

To calculate $P(Y,E)$ given a joint distribution table:

$$P(Y,E) = \sum_{W} P(Y,E,W)$$

The Chain rule can be used to factor out $P(Y,E,W)$.

Calculating $P(E)$ from the joint distribution table is redundant. Rather than calculating it from the whole table, one can use $P(Y,E)$ and marginalize on Y:

$$P(E) = \sum_{Y} P(Y,E)$$

Before marginalizing W, a joint distribution table could be constructed from the graph by multiplying all the factors. Using this method, of multiplying everything followed by marginalization, results in an exponentially large table which must then be marginalized upon. Hence, this method is not efficient. Instead, multiplying factors that share a certain Random variable $w \in W$, and then marginalizing upon the product of the factors would be more efficient as marginalization would be done on smaller CPTs.

This algorithm can be used in Figure 4, which is an extension of Figure 1, where G is a random variable representing whether the grass is wet (g_0 and g_1 mean the grass is dry and wet respectively). The algorithm must return $P(G = g_1|S = s_0)$ i.e. the probability of the grass being wet given no slip. It must create a CPT for $P(G|S)$. S belongs to E and G belongs to Y. R and W belong to W and must be marginalized.

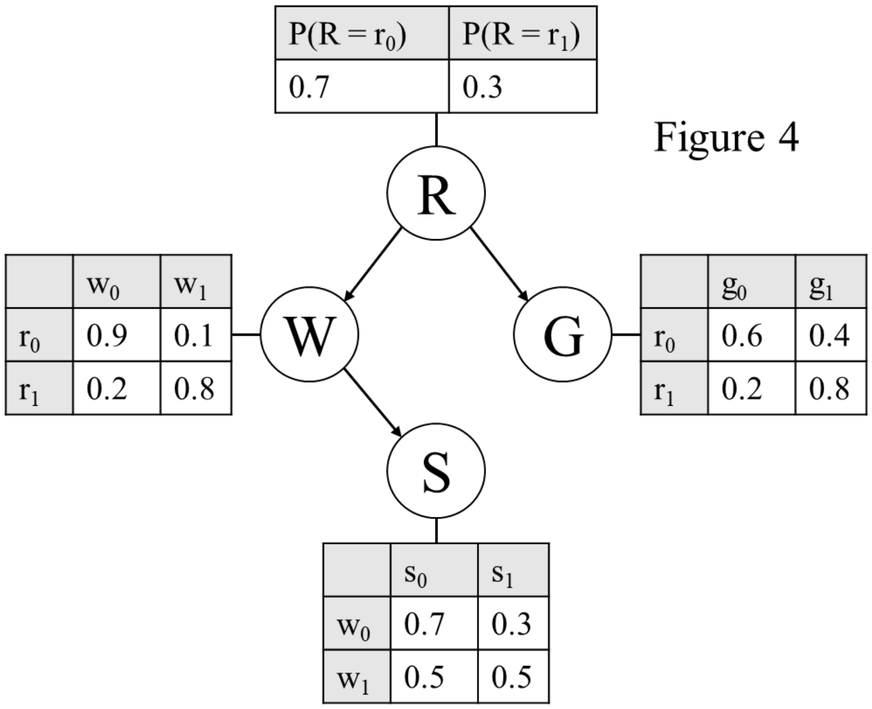


Figure 4

$P(R = r_0)$	$P(R = r_1)$
0.7	0.3

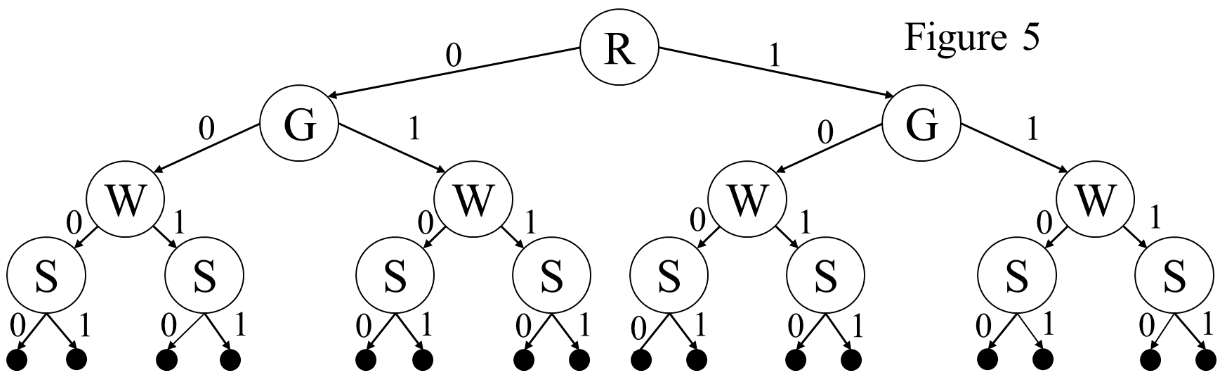
	w_0	w_1
r_0	0.9	0.1
r_1	0.2	0.8

	g_0	g_1
r_0	0.6	0.4
r_1	0.2	0.8

	s_0	s_1
w_0	0.7	0.3
w_1	0.5	0.5

The chance of grass being wet given no rain is higher than the chance of the pavement being wet given no rain as other factors such as dewdrops, and sprinklers can cause the grass to be wet.

To solve this using the Bayes theorem, I constructed a tree to help visualize which paths I would need to follow. For this example, only the paths with s_0 as the leaf nodes need to be considered. For the numerator in the Bayes theorem i.e. $P(g_1, s_0)$, only the paths with s_0 as the leaf and pass through g_1 need to be considered.



Given this tree, $P(G = g_1|S = s_0)$ can be calculated as shown:

$$P(G = g_1|S = s_0) = \frac{P(g_1, s_0)}{P(s_0)} = \frac{P(g_1, s_0)}{P(g_1, s_0) + P(g_0, s_0)}$$

$$P(g_1, s_0) = P(r_0, g_1, w_0, s_0) + P(r_0, g_1, w_1, s_0) + P(r_1, g_1, w_0, s_0) + P(r_1, g_1, w_1, s_0)$$

$P(r_0, g_1, w_0, s_0)$ can be calculated by factoring it out:

$$P(r_0, g_1, w_0, s_0) = P(r_0) \times P(g_1|r_0) \times P(w_0|r_0) \times P(s_0|w_0) = 0.7 \times 0.4 \times 0.9 \times 0.7 \approx 0.176$$

Therefore,

$$P(g_1, s_0) = 0.1764 + (0.7 \times 0.4 \times 0.1 \times 0.5) + (0.3 \times 0.8 \times 0.2 \times 0.7)$$
$$+ (0.3 \times 0.8 \times 0.8 \times 0.5) = 0.320$$

$P(g_0, s_0)$ can be calculated as shown:

$$P(r_0, g_0, w_0, s_0) + P(r_0, g_0, w_1, s_0) + P(r_1, g_0, w_0, s_0) + P(r_1, g_0, w_1, s_0)$$

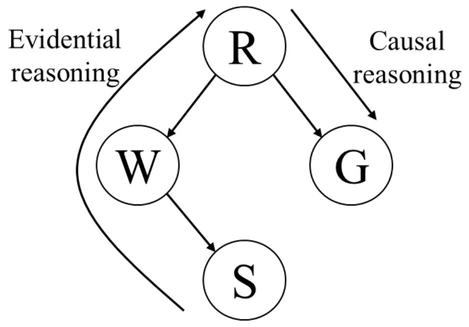
$$= (0.7 \times 0.6 \times 0.9 \times 0.7) + (0.7 \times 0.6 \times 0.1 \times 0.5) + (0.3 \times 0.2 \times 0.2 \times 0.7)$$
$$+ (0.3 \times 0.2 \times 0.8 \times 0.5) = 0.318$$

Finally,

$$P(G = g_1 | S = s_0) = \frac{0.32}{0.32 + 0.318} \approx 0.502$$

If *s₁* was conditioned upon, and the probability of *g₁* was to be calculated, the probability returned would be ~0.552, which is higher than 0.502. This makes sense as if one slips, the chances the pavement was wet is higher, meaning the chances it rained is higher, which means the chances the grass is wet is higher. The flow of influence is shown below:

Figure 6



Solving this problem using Variable elimination allows one to query any value for *S* and *G* as the entire CPT for $P(G|S)$ would be generated:

Factorizing $P(R, G, W, S)$ and marginalizing on *R* and *W* would result in:

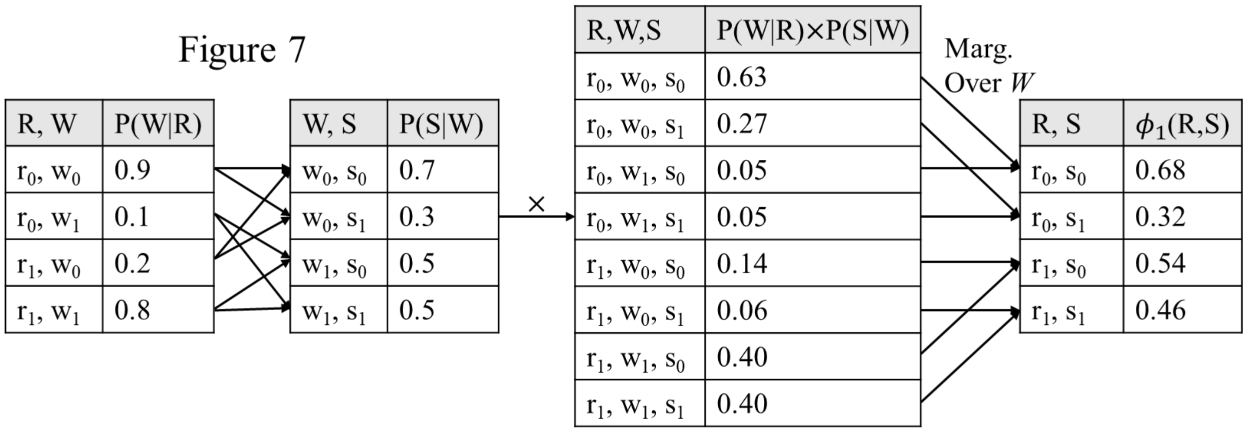
$$P(R, G, W, S) = \sum_{R,W} P(R) \times P(G|R) \times P(W|R) \times P(S|W)$$

First, marginalize *W*. Since $P(R)$ and $P(S|W)$ do not have *W* in its scope, they do not need to be in the sum:

$$P(G, S) = \sum_{R} P(R) \times P(G|R) \sum_{W} P(W|R) \times P(S|W)$$

This would result in $P(W|R)$ and $P(S|W)$ being multiplied, creating the factor ϕ_1:

Figure 7



R, W	P(W|R)
r_0, w_0	0.9
r_0, w_1	0.1
r_1, w_0	0.2
r_1, w_1	0.8

W, S	P(S|W)
w_0, s_0	0.7
w_0, s_1	0.3
w_1, s_0	0.5
w_1, s_1	0.5

×

R,W,S	P(W|R)×P(S|W)
r_0, w_0, s_0	0.63
r_0, w_0, s_1	0.27
r_0, w_1, s_0	0.05
r_0, w_1, s_1	0.05
r_1, w_0, s_0	0.14
r_1, w_0, s_1	0.06
r_1, w_1, s_0	0.40
r_1, w_1, s_1	0.40

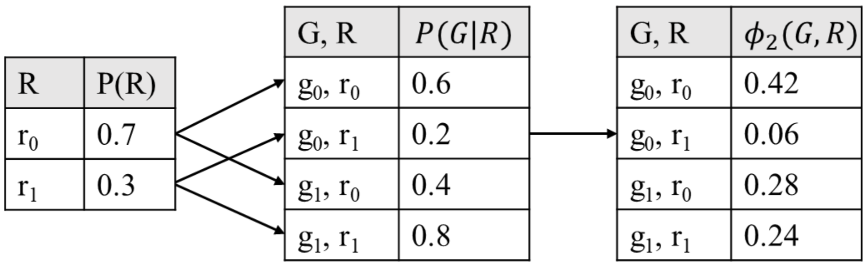
Marg. Over W

R, S	ϕ_1(R,S)
r_0, s_0	0.68
r_0, s_1	0.32
r_1, s_0	0.54
r_1, s_1	0.46

Now,

$$P(G,S) = \sum_R P(R) \times P(G|R) \times \phi_1(R,S)$$

Multiplying $P(R)$, $P(G|R)$ $\phi_1(R,S)$ would result in the following CPT for $\phi_3(G,S)$:

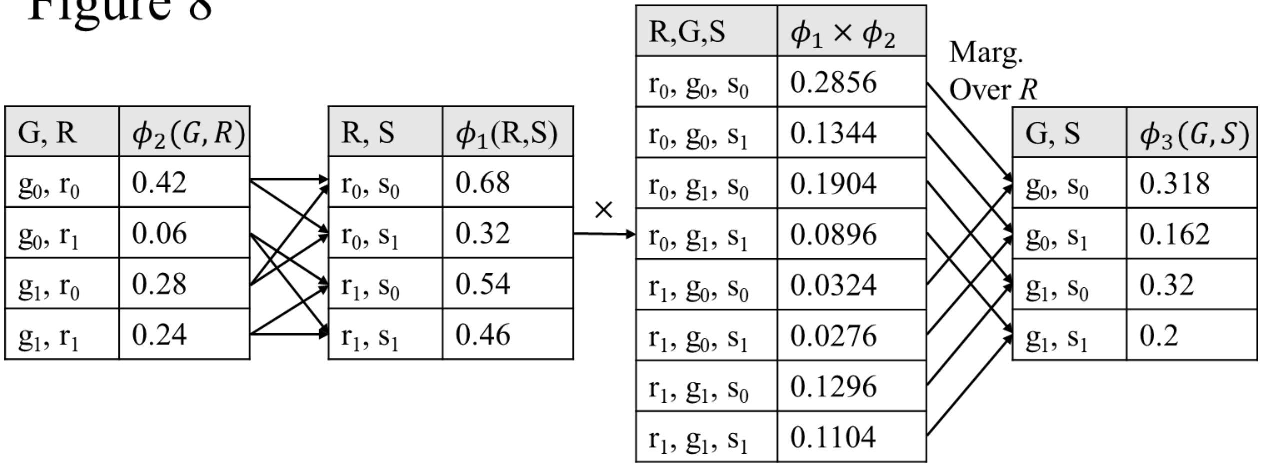


R	P(R)
r_0	0.7
r_1	0.3

G, R	$P(G\|R)$
g_0, r_0	0.6
g_0, r_1	0.2
g_1, r_0	0.4
g_1, r_1	0.8

G, R	$\phi_2(G,R)$
g_0, r_0	0.42
g_0, r_1	0.06
g_1, r_0	0.28
g_1, r_1	0.24

Figure 8



G, R	$\phi_2(G,R)$
g_0, r_0	0.42
g_0, r_1	0.06
g_1, r_0	0.28
g_1, r_1	0.24

R, S	ϕ_1(R,S)
r_0, s_0	0.68
r_0, s_1	0.32
r_1, s_0	0.54
r_1, s_1	0.46

×

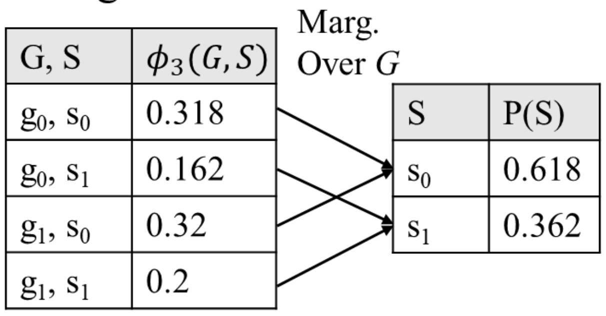
R,G,S	$\phi_1 \times \phi_2$
r_0, g_0, s_0	0.2856
r_0, g_0, s_1	0.1344
r_0, g_1, s_0	0.1904
r_0, g_1, s_1	0.0896
r_1, g_0, s_0	0.0324
r_1, g_0, s_1	0.0276
r_1, g_1, s_0	0.1296
r_1, g_1, s_1	0.1104

Marg. Over R

G, S	$\phi_3(G,S)$
g_0, s_0	0.318
g_0, s_1	0.162
g_1, s_0	0.32
g_1, s_1	0.2

Now that the CPT for $\phi_3(G,S)$ – or $P(G,S)$ – has been calculated, the CPT for $P(S)$ can be calculated by marginalizing over *G*:

Figure 9



G, S	$\phi_3(G,S)$
g_0, s_0	0.318
g_0, s_1	0.162
g_1, s_0	0.32
g_1, s_1	0.2

S	P(S)
s_0	0.618
s_1	0.362

Given these two CPTs, $P(G|S)$ can be calculated for any value of *G* and *S*. This can be compared to the values calculated through the usual way of using the Bayes theorem:

$P(G = g_1|S = s_0)$ would equal $\frac{P(g_1,s_0)}{P(s_0)}$. Substituting the values from the CPTs would result in the following probability: $\frac{0.32}{0.618} \approx 0.502$.

Likewise, to find $P(G = g_1|S = s_w)$, the values for $P(g_1, s_1)$ and $P(s_1)$ can be substituted, resulting in the following probability: $\frac{0.2}{0.362} \approx 0.552$.

Both these values match the probabilities previously calculated. Since the probabilities for $(g_0|s_0)$ and $(g_0|s_1)$ are complementary (to *1*) to $(g_1|s_0)$ and $(g_0|s_1)$ respectively, all the probabilities are correct. Though this method was time-consuming by hand, since variable elimination involves repeated matrix multiplication, and because computers can use parallel processing, this algorithm is much faster to run on computers.

Efficiency

Given that multiple algorithms exist for exact inference, it is important to assess the efficiency of Variable Elimination to conclude whether it is a useful algorithm [Koller, "Complexity of Variable Elimination"].

Variable elimination consists of two distinct steps: Multiplication, Addition (or Marginalization). These contribute to the runtime of this algorithm. Both multiplication and addition are constant time arithmetic operations as they are independent of the number of factors and the scope. To calculate the runtime, I first defined the variables.

In terms of the multiplication step, let m_k be the number of factors that share a common random variable $k \in W$. Let ϕ_k be the factor created by multiplying all factors that share k and let S_k represent the scope for this factor:

$$\phi_k(S_k) = \prod_{i=1}^{m_k} \phi_i$$

Next, let N_k equal the length of the CPT of ϕ_k. Considering m_k and N_k, the cost of multiplication per random variable in W would be $N_k(m_k - 1)$, as for each row in ϕ_k, there will be $m_k - 1$ multiplications. The cost uses $m_k - 1$, not m_k as multiplying two numbers counts as one and multiplying three counts as two and so on.

In terms of the marginalization step, let m be the number of factors starting with. In BN, m is at most n where n is the total number of random variables in the network. At each addition or marginalization step, a new factor is created, and one random variable is eliminated. This means there can be a maximum of n elimination steps and new factors created. Therefore, the total number of factors to exist in variable elimination, m_T¸ must equal $m + n$. The number of sum operations in each elimination step would be upper bounded by N_k.

The total number of multiplications in variable elimination would be $\sum_k^W (m_k - 1)N_k$, as for each random variable, the number of multiplications is $N_k(m_k - 1)$ and there are k factors

Each distinct factor is only multiplied in only once to create a new factor. Therefore, $\sum_k^W (m_k - 1)$ is upper bounded by *m'*. Let *N* equal the largest factor created over all multiplication steps i.e. $\max(N_k)$ for all k. Hence, the total number of product operations would be upper bounded by $N \times m'$.

The total number of sum operations in variable elimination would be:

$$\sum_k^W N_k$$

Since *N* is the largest CPT created from multiplication, and because there is a total of *n* elimination steps, $\sum_k^W N_k$ would be upper bounded by $N \times n$.

These two runtimes suggest that the total runtime of Variable elimination is linear in relation to *N*, *m'*, and *n*. However, the algorithm as a whole is not linear.

Let *d* be the maximum number of distinct values a random variable has in the BN and let *r* be the maximum number of variables in the scope of a factor in the BN. Here, *N* would be d^r. meaning the size of the CPT resulting from the multiplication step would be exponential in relation to the cardinality of the scope of the ith factor.

Therefore, variable elimination has an exponential runtime. Exponential runtimes are impractical for algorithms in general as even for relatively small input sizes (< 100), the runtime is high.

A graphical approach to Variable Elimination

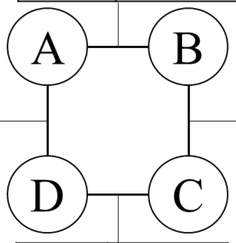
By using a different type of graph called ‘Markov Networks’, an alternative approach can be used to eliminate variables. It is upon this approach that algorithms for deciding the order in which variables are eliminated are based [Koller, “Graph-based Perspective on Variable Elimination”].

Markov networks are undirected probabilistic graph models. Like BN, nodes in Markov networks represent a random variable. Unlike BNs, edges in a Markov network represent an association between random variables, not necessarily a probabilistic influence. Each edge or group of edges represents a factor. Figure 10 is an example of a Markov network representing random variables *A*¸ *B, C,* and *D*. Each random variable takes binary values, and the return value of the factors represents an ‘affinity’ score between the variables it takes in for certain values.

Figure 10

A, B	ϕ_1(A, B)
a_0, b_0	10
a_0, b_1	1
a_1, b_0	5
a_1, b_1	9

D, A	ϕ_4(D, A)
d_0, a_0	6
d_0, a_1	8
d_1, a_0	7
d_1, a_1	8



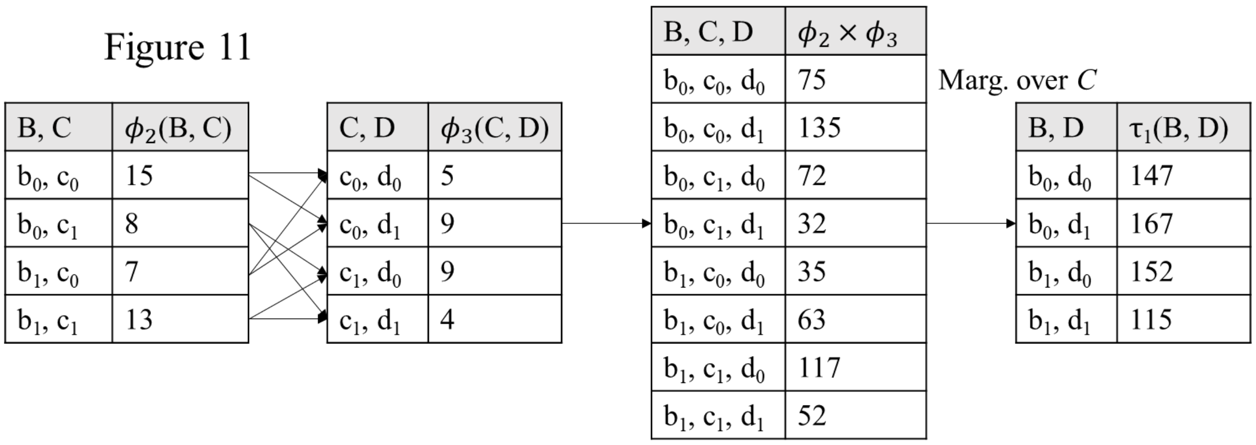
B, C	ϕ_2(B, C)
b_0, c_0	15
b_0, c_1	8
b_1, c_0	7
b_1, c_1	13

C, D	ϕ_3(C, D)
c_0, d_0	5
c_0, d_1	9
c_1, d_0	9
c_1, d_1	4

Variable elimination in Markov networks is like Variable elimination in BNs: factors with a common random variable *w* belonging to *W* are multiplied and have *w* marginalized. However, a factor describing a set of random variables *s* must have all variables in *s* directly connected by an edge. Therefore, when a new factor with a different scope is created from multiplying two or more factors, the random variables belonging to the scope of the new factor must all be connected.

In Figure 10, if *C* were to be eliminated, ϕ_2 and ϕ_3 must be multiplied as they are the only two factors with *C* in their scope. Then, *C* must be marginalized. The new factor, τ_1, would have a scope of *B* and *D*. Hence, a new edge would need to be created to connect *B* and *D*.

Figure 11



B, C	ϕ_2(B, C)
b_0, c_0	15
b_0, c_1	8
b_1, c_0	7
b_1, c_1	13

C, D	ϕ_3(C, D)
c_0, d_0	5
c_0, d_1	9
c_1, d_0	9
c_1, d_1	4

B, C, D	$\phi_2 \times \phi_3$
b_0, c_0, d_0	75
b_0, c_0, d_1	135
b_0, c_1, d_0	72
b_0, c_1, d_1	32
b_1, c_0, d_0	35
b_1, c_0, d_1	63
b_1, c_1, d_0	117
b_1, c_1, d_1	52

B, D	τ_1(B, D)
b_0, d_0	147
b_0, d_1	167
b_1, d_0	152
b_1, d_1	115

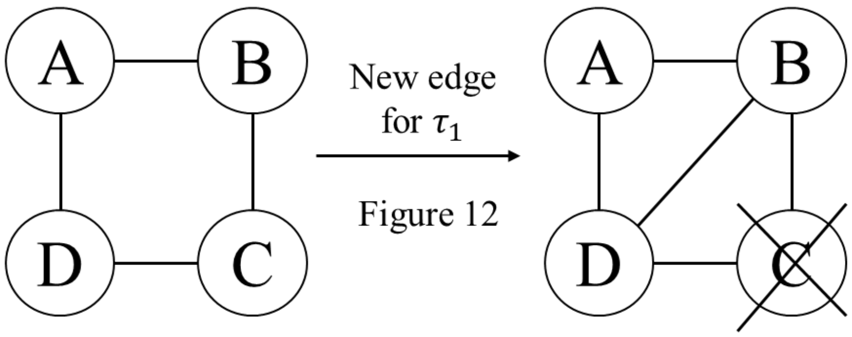
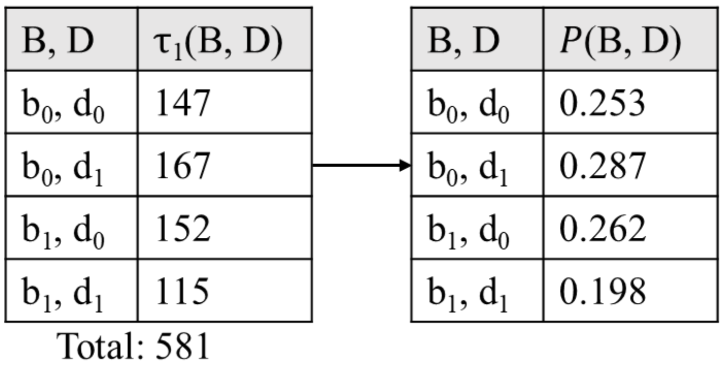


Figure 12

The values of τ_1 can then be normalized to calculate a probability distribution:

Figure 13



B, D	τ_1(B, D)
b_0, d_0	147
b_0, d_1	167
b_1, d_0	152
b_1, d_1	115

Total: 581

B, D	*P*(B, D)
b_0, d_0	0.253
b_0, d_1	0.287
b_1, d_0	0.262
b_1, d_1	0.198

Renormalizing MNs is done by summing all values in the CPT to get normalizing constant *Z*, and dividing each value by *Z*.

The modified graph created from eliminating variables is called the ‘Induced graph’.

A ‘Clique’ refers to a sub-graph (a collection of nodes in a graph) where all nodes are connected (i.e. they are adjacent) and no other nodes can be added to increase the size of the clique. In Figure 12, the graph, before eliminating *C*, has four cliques: {*A*, *B*}, {*B*, *C*}, {*C*, *D*}, {*D*, *A*}. However, in the induced graph, there is only one clique: {*A*, *B*, *D*}.

The ‘Induced width’ refers to the largest clique in the graph. It corresponds to the runtime of variable elimination as a larger clique will require a factor with a scope with a higher cardinality. The size of the CPT created by multiplying the factors (N_k) is exponential in relation to the cardinality of the scope of the factor. The factor describing all variables in the largest clique will also have the largest scope. Hence, the size of the CPT is exponential in relation to the induced width as well.

BNs can be converted to Markov networks. However, the induced edges must be drawn to satisfy each factor's scope. Figure 14 is an example of an arbitrary BN being converted to a Markov network.

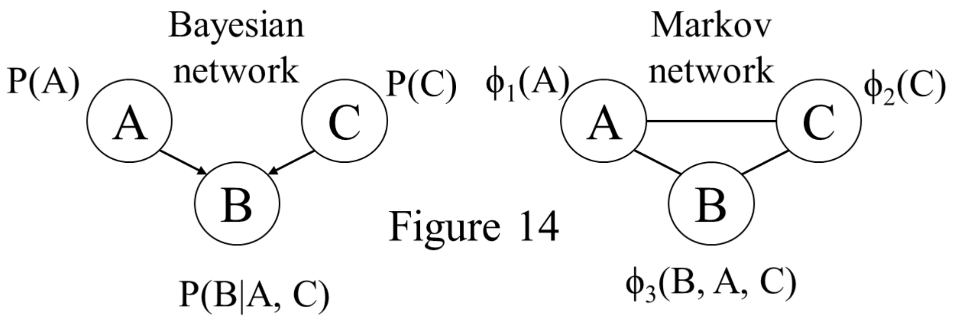


Figure 14

Variable elimination orderings

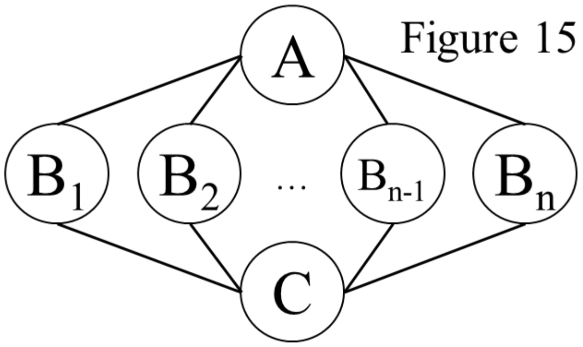
Finding the optimal variable elimination ordering is an NP-Complete problem. However, better elimination orderings can be found using 'Greedy' algorithms. Greedy algorithms locally optimize. Given a step and choices, the algorithm will choose the best choice at that moment without considering future consequences. This means Greedy algorithms will not always give the optimal solution.

There are many greedy algorithms to determine a variable elimination ordering. This part of the essay will discuss two of them:

1. Number of neighbors: start by eliminating the random variable that belongs to *W* that has the least number of neighbors.
2. Number of induced edges created: start by eliminating the random variable that belongs to *W* that, when eliminated, results in the fewest number of new edges being created.

There is no definitive way to rank the algorithms in terms of the improvement of efficiency resulting from them as their efficiency varies from graph to graph. The algorithm based on the number of neighbors may be better in one graph, while the algorithm based on the number of induced edges created may be better on others.

In the case of Figure 15, eliminating Node *A* first would result in a factor ϕ, with a scope of $\phi(B_1, B_2, \ldots, B_{n-1}, B_n)$. This factor's scope has a cardinality *n*. If the maximum number of distinct values a random variable in the factor has is *d*, the size of the CPT after multiplying the *n* factors in is upper bounded d^n. The factor size is exponential in *n*.



Eliminating *Bi* before *A* or *C* is more efficient. By eliminating *B1* first, two factors will be multiplied in:

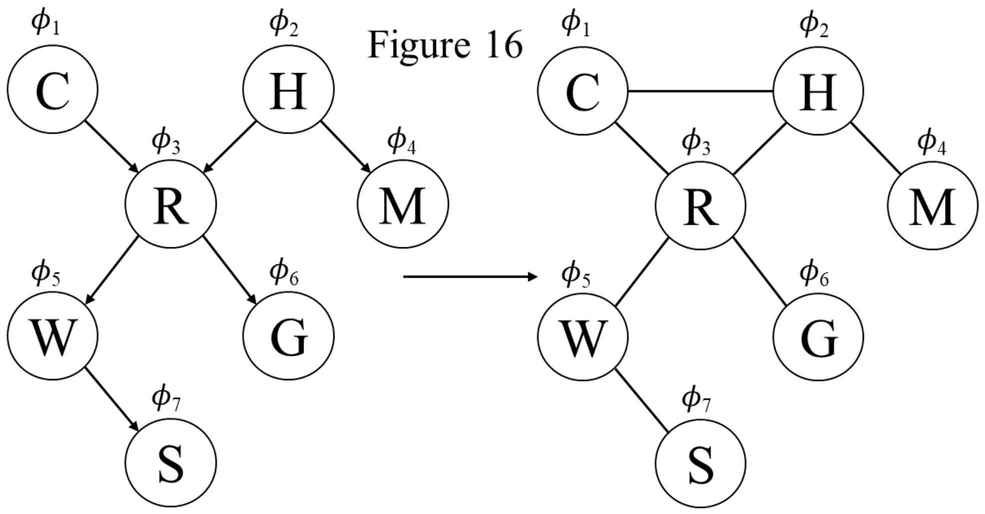
$$\phi_1(A, B_1) \times \phi_2(B_1, C) = (\phi_1 \times \phi_2)(A, B_1, C) \xrightarrow{marginalize\ on\ B_1} \tau_1(A, C)$$

This is repeated for all B_i $\{1 \leq i \leq n\}$, creating the set of factors $\tau_i(A, C)$. To find a probability distribution for a factor of *A* and *C*, all τ_i should be multiplied in and the resulting factor should be renormalized:

$$P(A, C) = \frac{1}{Z} \times \prod_{i=1}^{n} \tau_i(A, C)$$

Since properties such as neighbors are unique to Markov networks, a BN will need to be converted to a Markov network for the first and third greedy algorithms to work. Figure 16 is a

BN that is an extension of figure 4. *C* refers to whether it is cloudy or not (*c₀*: not cloudy, *c₁*: cloudy). *H* refers to whether it is humid or not (*h₀*: not humid, *h₁*: humid). *M* refers to whether mist is present or not (*m₀*: not misty, *m₁*: misty). The CPTs for the factors can be found in Appendix. A. Figure 16 also shows the BN being converted to a Markov network.



Using the chain rule, the graph can be factorized as followed:

$$\phi(C,H,R,M,W,G,S) = \phi_1(C)\phi_2(H)\phi_3(R,C,H)\phi_4(M,H)\phi_5(W,R)\phi_6(G,R)\phi_7(S,W)$$

Suppose $P(M,S)$ was to be found. To do so, all other variables would have to be marginalized:

$$\phi(M,S) = \sum_{C,H,R,W,G} \phi_1(C)\phi_2(H)\phi_3(R,C,H)\phi_4(M,H)\phi_5(W,R)\phi_6(G,R)\phi_7(S,W)$$

To start, the variables will be eliminated in the following worst-case ordering: $\{R,H,W,C,G\}$, which is the reverse of the first greedy algorithm:

$$\phi(M,S) = \sum_{C,H,W,G} \phi_1(C)\phi_2(H)\phi_7(S,W)\phi_4(M,H)\sum_{R}\phi_3(R,C,H)\phi_5(W,R)\phi_6(G,R)$$

Let $\tau_1(C,H,W,G) = \sum_R \phi_3(R,C,H)\phi_5(W,R)\phi_6(G,R)$ [Appendix. B]

$$\phi(M,S) = \sum_{C,W,G} \phi_1(C)\phi_7(S,W) \sum_{H} \phi_2(H)\phi_4(M,H)\tau_1(C,H,W,G)$$

Let $\tau_2(C,W,G,M) = \sum_H \phi_2(H)\phi_4(M,H)\tau_1(C,H,W,G)$

$$\phi(M,S) = \sum_{C,G} \phi_1(C) \sum_{W} \phi_7(S,W)\tau_2(C,W,G,M)$$

Let $\tau_3(S,C,G,M) = \sum_W \phi_7(S,W)\tau_2(C,W,G,M)$

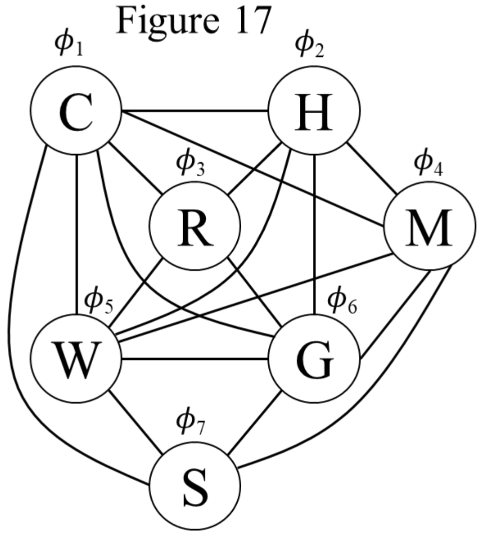
$$\phi(M,S) = \sum_{G}\sum_{C} \phi_1(C)\tau_3(S,C,G,M)$$

Let $\tau_4(S,G,M) = \sum_C \phi_1(C)\tau_3(S,C,G,M)$

$$\phi(M,S) = \sum_{G} \tau_4(S,G,M)$$

The induced graph using this method would look as follows:

Figure 17



The induced width of this graph is *5*, as there is no clique having more than five random variables. Therefore, the largest CPT created would be $2^5 = 32$. The CPTs calculated also support this. It's important to recognize that a 32-length-CPT is present in every marginalization step but the last one as there are multiple width-5-cliques.

Next, the variables will be eliminated in the order determined by the first greedy algorithm: $\{G, C, W, H, R\}$.

$$\phi(M,S) = \sum_{C,H,R,W} \phi_1(C)\phi_2(H)\phi_3(R,C,H)\phi_4(M,H)\phi_5(W,R)\phi_7(S,W) \sum_{G} \phi_6(G,R)$$

Let $\tau_1(R) = \sum_G \phi_6(G,R)$ [Appendix C.1]

$$\phi(M,S) = \sum_{H,R,W} \tau_1(R)\phi_2(H)\phi_4(M,H)\phi_5(W,R)\phi_7(S,W) \sum_{C} \phi_1(C)\phi_3(R,C,H)$$

Let $\tau_2(R,H) = \sum_C \phi_1(C)\phi_3(R,C,H)$ [Appendix C.2]

$$\phi(M,S) = \sum_{H,R} \tau_1(R)\phi_2(H)\phi_4(M,H)\tau_2(R,H) \sum_{W} \phi_5(W,R)\phi_7(S,W)$$

Let $\tau_3(R,S) = \sum_W \phi_5(W,R)\phi_7(S,W)$ [Appendix C.3]

$$\phi(M,S) = \sum_{R} \tau_1(R)\tau_3(R,S) \sum_{H} \phi_2(H)\phi_4(M,H)\tau_2(R,H)$$

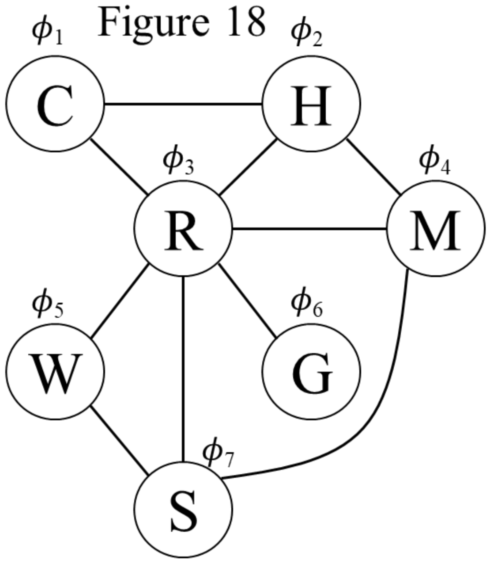
Let $\tau_4(R,M) = \sum_H \phi_2(H)\phi_4(M,H)\tau_2(R,H)$ [Appendix C.4]

$$\phi(M,S) = \sum_{R} \tau_1(R)\tau_3(R,S)\tau_4(R,M)$$

[Appendix C.5]

Using this ordering, the induced graph would look as follows:

Figure 18



The graph has an induced width of three, as no clique is larger than three. Therefore, the largest factor created during multiplication is of length 2^3, or 8. This is supported by the CPTs.

Using the second greedy algorithm, the ordering of elimination would be: $\{G, C, W, H, R\}$, which is identical to the ordering of the previous algorithm. Hence, the largest CPT created here would also be of length 8.

Evaluation

Since these two methods of deciding the ordering of variable elimination are greedy algorithms, one cannot be definitively better than the other, as some may perform better than others in certain graphs. However, as seen with these examples using a greedy algorithm can decrease the induced width, therefore, ensuring the algorithm is more efficient.

Conclusion

Through this essay, I discovered how variable elimination is a valid method for Inference in Probabilistic graphic models through two approaches. The first involves using Bayesian Networks and the second involves Markov Networks. I also learned how VE has an exponential runtime in relation to the number of random variables and is hence not an efficient algorithm. However, this efficiency can be improved using greedy algorithms.

Appendices

A

C	ϕ_1: P(C)
c_0	0.6
c_1	0.4

H	ϕ_2: P(H)
h_0	0.5
h_1	0.5

M, H	ϕ_4: P(M|H)
m_0, h_0	0.9
m_0, h_1	0.4
m_1, h_0	0.1
m_1, h_1	0.6

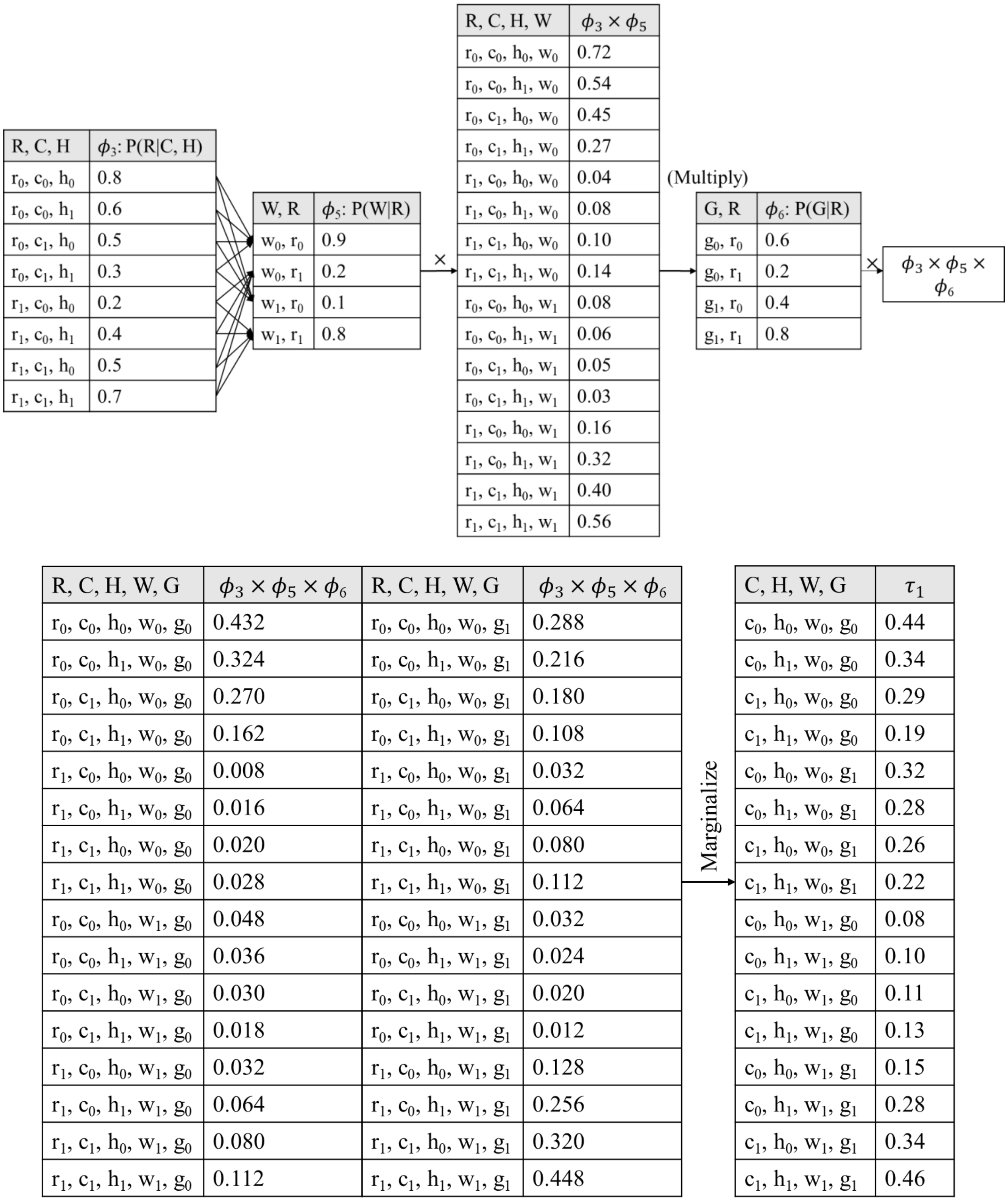
R, C, H	ϕ_3: P(R|C, H)
r_0, c_0, h_0	0.8
r_0, c_0, h_1	0.6
r_0, c_1, h_0	0.5
r_0, c_1, h_1	0.3
r_1, c_0, h_0	0.2
r_1, c_0, h_1	0.4
r_1, c_1, h_0	0.5
r_1, c_1, h_1	0.7

W, R	ϕ_5: P(W|R)
w_0, r_0	0.9
w_0, r_1	0.2
w_1, r_0	0.1
w_1, r_1	0.8

S, W	ϕ_7: P(S|W)
s_0, w_0	0.7
s_0, w_1	0.5
s_1, w_0	0.3
s_1, w_1	0.5

G, R	ϕ_6: P(G|R)
g_0, r_0	0.6
g_0, r_1	0.2
g_1, r_0	0.4
g_1, r_1	0.8

B



R, C, H	ϕ_3: P(R|C, H)
r_0, c_0, h_0	0.8
r_0, c_0, h_1	0.6
r_0, c_1, h_0	0.5
r_0, c_1, h_1	0.3
r_1, c_0, h_0	0.2
r_1, c_0, h_1	0.4
r_1, c_1, h_0	0.5
r_1, c_1, h_1	0.7

W, R	ϕ_5: P(W|R)
w_0, r_0	0.9
w_0, r_1	0.2
w_1, r_0	0.1
w_1, r_1	0.8

R, C, H, W	$\phi_3 \times \phi_5$
r_0, c_0, h_0, w_0	0.72
r_0, c_0, h_1, w_0	0.54
r_0, c_1, h_0, w_0	0.45
r_0, c_1, h_1, w_0	0.27
r_1, c_0, h_0, w_0	0.04
r_1, c_0, h_1, w_0	0.08
r_1, c_1, h_0, w_0	0.10
r_1, c_1, h_1, w_0	0.14
r_0, c_0, h_0, w_1	0.08
r_0, c_0, h_1, w_1	0.06
r_0, c_1, h_0, w_1	0.05
r_0, c_1, h_1, w_1	0.03
r_1, c_0, h_0, w_1	0.16
r_1, c_0, h_1, w_1	0.32
r_1, c_1, h_0, w_1	0.40
r_1, c_1, h_1, w_1	0.56

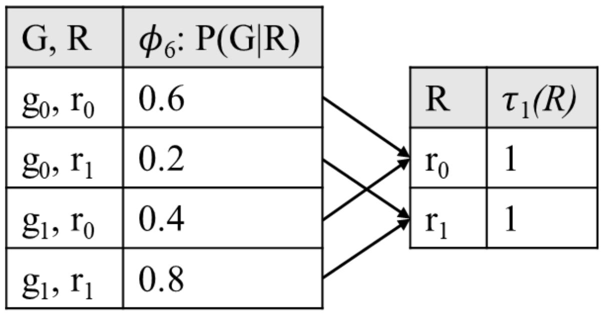
G, R	ϕ_6: P(G|R)
g_0, r_0	0.6
g_0, r_1	0.2
g_1, r_0	0.4
g_1, r_1	0.8

R, C, H, W, G	$\phi_3 \times \phi_5 \times \phi_6$	R, C, H, W, G	$\phi_3 \times \phi_5 \times \phi_6$
r_0, c_0, h_0, w_0, g_0	0.432	r_0, c_0, h_0, w_0, g_1	0.288
r_0, c_0, h_1, w_0, g_0	0.324	r_0, c_0, h_1, w_0, g_1	0.216
r_0, c_1, h_0, w_0, g_0	0.270	r_0, c_1, h_0, w_0, g_1	0.180
r_0, c_1, h_1, w_0, g_0	0.162	r_0, c_1, h_1, w_0, g_1	0.108
r_1, c_0, h_0, w_0, g_0	0.008	r_1, c_0, h_0, w_0, g_1	0.032
r_1, c_0, h_1, w_0, g_0	0.016	r_1, c_0, h_1, w_0, g_1	0.064
r_1, c_1, h_0, w_0, g_0	0.020	r_1, c_1, h_0, w_0, g_1	0.080
r_1, c_1, h_1, w_0, g_0	0.028	r_1, c_1, h_1, w_0, g_1	0.112
r_0, c_0, h_0, w_1, g_0	0.048	r_0, c_0, h_0, w_1, g_1	0.032
r_0, c_0, h_1, w_1, g_0	0.036	r_0, c_0, h_1, w_1, g_1	0.024
r_0, c_1, h_0, w_1, g_0	0.030	r_0, c_1, h_0, w_1, g_1	0.020
r_0, c_1, h_1, w_1, g_0	0.018	r_0, c_1, h_1, w_1, g_1	0.012
r_1, c_0, h_0, w_1, g_0	0.032	r_1, c_0, h_0, w_1, g_1	0.128
r_1, c_0, h_1, w_1, g_0	0.064	r_1, c_0, h_1, w_1, g_1	0.256
r_1, c_1, h_0, w_1, g_0	0.080	r_1, c_1, h_0, w_1, g_1	0.320
r_1, c_1, h_1, w_1, g_0	0.112	r_1, c_1, h_1, w_1, g_1	0.448

C, H, W, G	τ_1
c_0, h_0, w_0, g_0	0.44
c_0, h_1, w_0, g_0	0.34
c_1, h_0, w_0, g_0	0.29
c_1, h_1, w_0, g_0	0.19
c_0, h_0, w_0, g_1	0.32
c_0, h_1, w_0, g_1	0.28
c_1, h_0, w_0, g_1	0.26
c_1, h_1, w_0, g_1	0.22
c_0, h_0, w_1, g_0	0.08
c_0, h_1, w_1, g_0	0.10
c_1, h_0, w_1, g_0	0.11
c_1, h_1, w_1, g_0	0.13
c_0, h_0, w_1, g_1	0.15
c_0, h_1, w_1, g_1	0.28
c_1, h_0, w_1, g_1	0.34
c_1, h_1, w_1, g_1	0.46

C

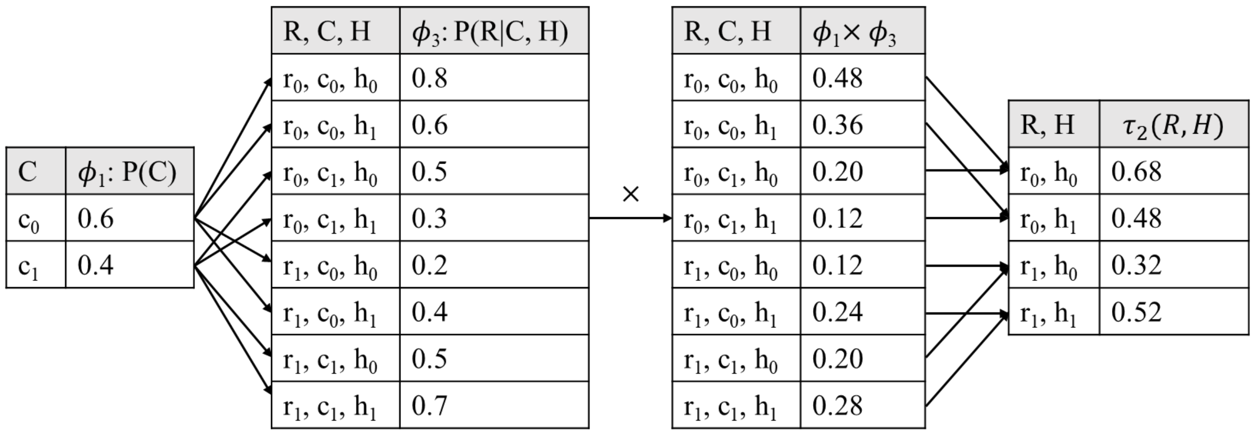
C.1



G, R	ϕ_6: P(G|R)
g_0, r_0	0.6
g_0, r_1	0.2
g_1, r_0	0.4
g_1, r_1	0.8

R	$\tau_1(R)$
r_0	1
r_1	1

C.2



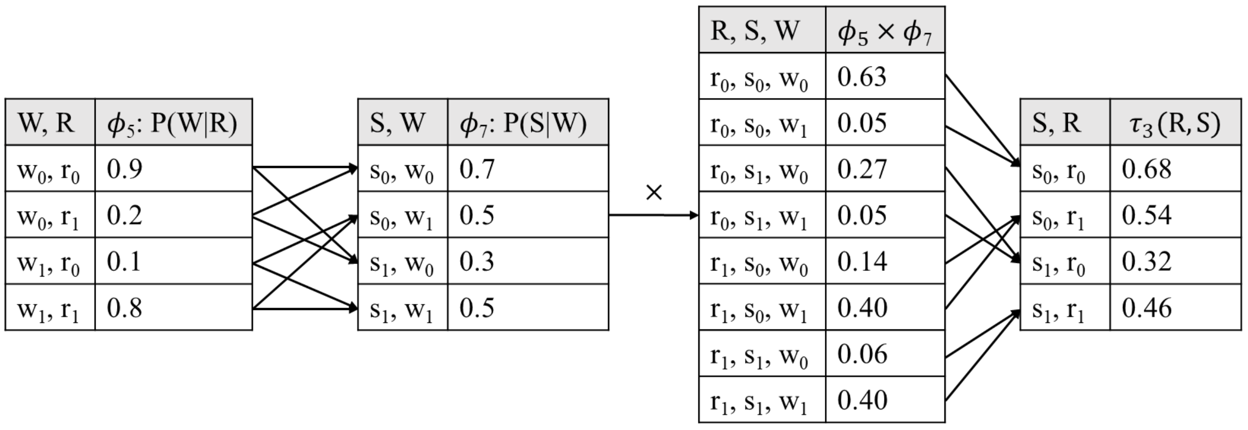
C	ϕ_1: P(C)
c_0	0.6
c_1	0.4

R, C, H	ϕ_3: P(R|C, H)
r_0, c_0, h_0	0.8
r_0, c_0, h_1	0.6
r_0, c_1, h_0	0.5
r_0, c_1, h_1	0.3
r_1, c_0, h_0	0.2
r_1, c_0, h_1	0.4
r_1, c_1, h_0	0.5
r_1, c_1, h_1	0.7

R, C, H	$\phi_1 \times \phi_3$
r_0, c_0, h_0	0.48
r_0, c_0, h_1	0.36
r_0, c_1, h_0	0.20
r_0, c_1, h_1	0.12
r_1, c_0, h_0	0.12
r_1, c_0, h_1	0.24
r_1, c_1, h_0	0.20
r_1, c_1, h_1	0.28

R, H	$\tau_2(R, H)$
r_0, h_0	0.68
r_0, h_1	0.48
r_1, h_0	0.32
r_1, h_1	0.52

C.3



W, R	ϕ_5: P(W|R)
w_0, r_0	0.9
w_0, r_1	0.2
w_1, r_0	0.1
w_1, r_1	0.8

S, W	ϕ_7: P(S|W)
s_0, w_0	0.7
s_0, w_1	0.5
s_1, w_0	0.3
s_1, w_1	0.5

R, S, W	$\phi_5 \times \phi_7$
r_0, s_0, w_0	0.63
r_0, s_0, w_1	0.05
r_0, s_1, w_0	0.27
r_0, s_1, w_1	0.05
r_1, s_0, w_0	0.14
r_1, s_0, w_1	0.40
r_1, s_1, w_0	0.06
r_1, s_1, w_1	0.40

S, R	τ_3(R, S)
s_0, r_0	0.68
s_0, r_1	0.54
s_1, r_0	0.32
s_1, r_1	0.46

C.4

H	ϕ_2: P(H)
h_0	0.5
h_1	0.5

M, H	ϕ_4: P(M|H)
m_0, h_0	0.9
m_0, h_1	0.4
m_1, h_0	0.1
m_1, h_1	0.6

$\times$

M, H	$\phi_2 \times \phi_4$
m_0, h_0	0.45
m_0, h_1	0.20
m_1, h_0	0.05
m_1, h_1	0.30

M, H	$\phi_2 \times \phi_4$
m_0, h_0	0.45
m_0, h_1	0.20
m_1, h_0	0.05
m_1, h_1	0.30

R, H	$\tau_2(R,H)$
r_0, h_0	0.68
r_0, h_1	0.48
r_1, h_0	0.32
r_1, h_1	0.52

$\times$

R, H, M	$\phi_2 \times \phi_4 \times \tau_2$
r_0, h_0, m_0	0.306
r_0, h_0, m_1	0.034
r_0, h_1, m_0	0.096
r_0, h_1, m_1	0.144
r_1, h_0, m_0	0.144
r_1, h_0, m_1	0.016
r_1, h_1, m_0	0.104
r_1, h_1, m_1	0.156

R, M	$\tau_4(R,M)$
r_0, m_0	0.402
r_0, m_1	0.178
r_1, m_0	0.248
r_1, m_1	0.172

C.5

R	$\tau_1(R)$
r_0	1
r_1	1

S, R	τ_3(R, S)
s_0, r_0	0.68
s_0, r_1	0.54
s_1, r_0	0.32
s_1, r_1	0.46

$\times$

S, R	$\tau_1 \times \tau_3$
s_0, r_0	0.68
s_0, r_1	0.54
s_1, r_0	0.32
s_1, r_1	0.46

R, M	$\tau_4(R,M)$
r_0, m_0	0.402
r_0, m_1	0.178
r_1, m_0	0.248
r_1, m_1	0.172

$\times$

R, S, M	$\tau_1 \times \tau_3 \times \tau_4$
r_0, s_0, m_0	0.27336
r_0, s_0, m_1	0.12104
r_0, s_1, m_0	0.12864
r_0, s_1, m_1	0.05686
r_1, s_0, m_0	0.13392
r_1, s_0, m_1	0.09288
r_1, s_1, m_0	0.11408
r_1, s_1, m_1	0.07912

R, S, M	$\tau_1 \times \tau_3 \times \tau_4$
r_0, s_0, m_0	0.27336
r_0, s_0, m_1	0.12104
r_0, s_1, m_0	0.12864
r_0, s_1, m_1	0.05686
r_1, s_0, m_0	0.13392
r_1, s_0, m_1	0.09288
r_1, s_1, m_0	0.11408
r_1, s_1, m_1	0.07912

M, S	$P(M,S)$
m_0, s_0	0.40728
m_0, s_1	0.24272
m_1, s_0	0.21392
m_1, s_1	0.13598

Works Cited

Huang, Bert. “Bayesian Networks - Youtube.” *Youtube*, 25 Mar. 2015, https://www.youtube.com/watch?v=TuGDMj43ehw.

Koller, Daphne. “Complexity of Variable Elimination - Variable Elimination.” *Coursera*, https://www.coursera.org/learn/probabilistic-graphical-models-2-inference/lecture/HaBqG/complexity-of-variable-elimination.

Koller, Daphne. “Flow of Probabilistic Influence - Bayesian Network (Directed Models).” *Coursera*, https://www.coursera.org/learn/probabilistic-graphical-models/lecture/1eCp1/flow-of-probabilistic-influence.

Koller, Daphne. “Graph-Based Perspective on Variable Elimination - Variable Elimination.” *Coursera*, https://www.coursera.org/learn/probabilistic-graphical-models-2-inference/lecture/tAtMr/graph-based-perspective-on-variable-elimination.

Malone, Brandon. *Factor Elimination - CS.HELSINKI.FI*. https://www.cs.helsinki.fi/u/bmmalone/probabilistic-models-spring-2014/FactorElimination.pdf.

Bibliography

"Causality." *Wikipedia*, Wikimedia Foundation, 2 Dec. 2021, https://en.wikipedia.org/wiki/Causality.

"Definition of BBNs: Graphs and Probability Tables." *Definition of Bbns: Graphs and Probability Tables*, https://www.eecs.qmul.ac.uk/~norman/BBNs/Definition_of_BBNs__graphs_and_probability_tables.htm.

Independence and Conditional Independence, https://www.eecs.qmul.ac.uk/~norman/BBNs/Independence_and_conditional_independence.htm.

Pishro-Nik, Hossein. *Conditional Independence*, https://www.probabilitycourse.com/chapter1/1_4_4_conditional_independence.php#:~:text=Definition,P(B)%3E0.