Lecture 9:

Exact Inference

Problems - highly dependent data

Data Distribution

a0,b3,c3,d1

a1,b1,c2,d3

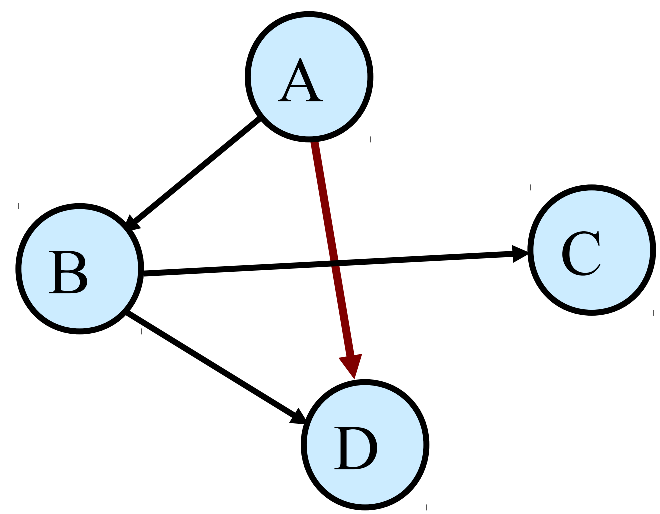
a1,b2, c1,d2

&c.

&c.

Find most dependent arcs that form a tree

Variables



Probability propagation can be difficult

Problems - highly dependent data

Data Distribution

a0,b3,c3,d1

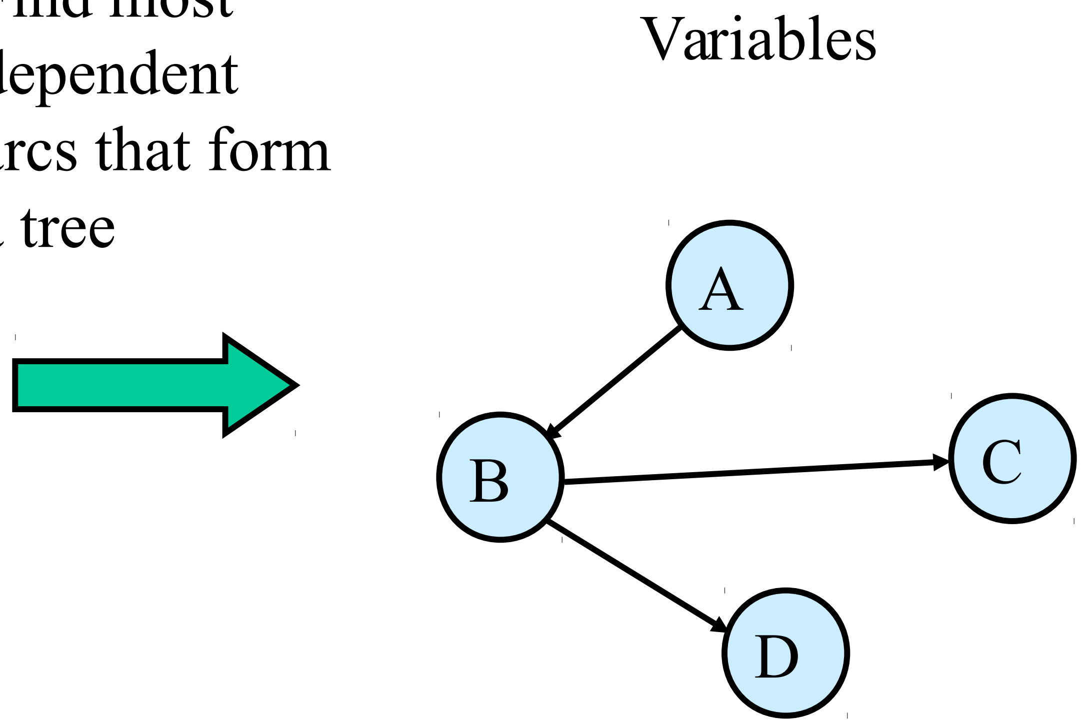
a1,b1,c2,d3

a1,b2, c1,d2

&c.

&c.

Find most dependent arcs that form a tree



The network does not exactly represent the data!

Methods of Exact Inference

Exact Inference means modelling all the dependencies in the data.

This is only computationally feasible for relatively small or sparse networks.

Methods:

- Cutset Conditioning (Pearl)
- Node Clustering
- Join Trees (Lauritizen and Speigelhalter)

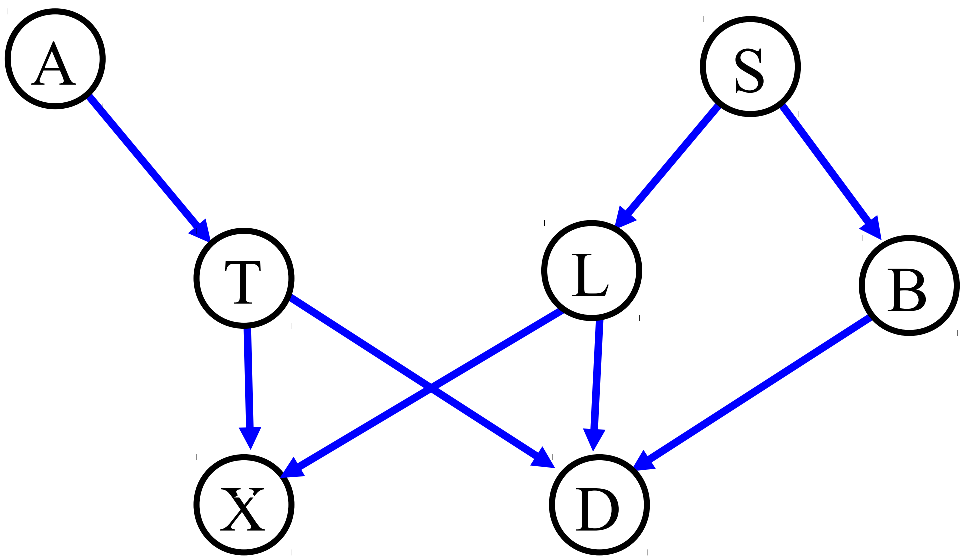
Cutset Conditioning

Pearl suggested a method for propagating probabilities in multiply connected networks.

The first step is to identify a cut set of nodes, which if instantiated makes propagation terminate correctly.

Cutset Conditioning Example 1

Given the Asia network, the minimum cutset consists of node L. If L is instantiated propagation is safe.

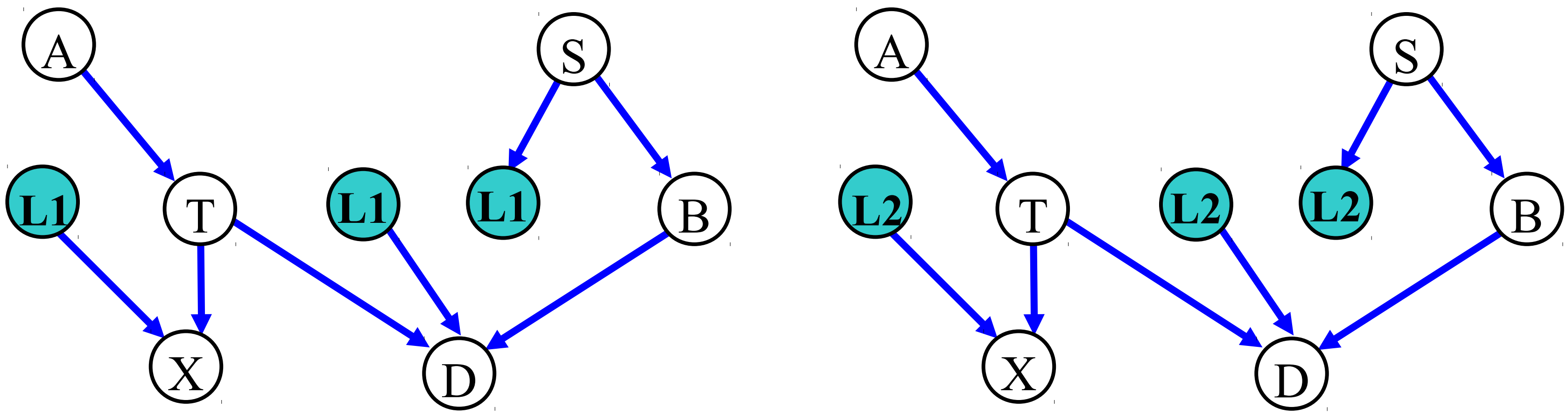


A : Asia
B : Bronchitis
D : Dyspnea
L : Lung Cancer
S : Smoking
T : Tuberculosis
X : XRay (Positive)

Cutset Conditioning Example 2

If data is not available on L the network can be broken into several networks, one for each possible instantiated state of L.

L has 2 states and both networks can be treated as if singly connected:



Cutset Conditioning Example 3

Probabilities can be propagated correctly in both networks.

A final value for the probability can be found by weighting the results according to the prior probability distribution for L.

Problems with cutset conditioning 1

The method relies on finding a small cutset.

If the cutset consists of n nodes each with t states then the probability propagations need to be carried out t^n times.

We see that the computation time expands exponentially - it is n-p hard

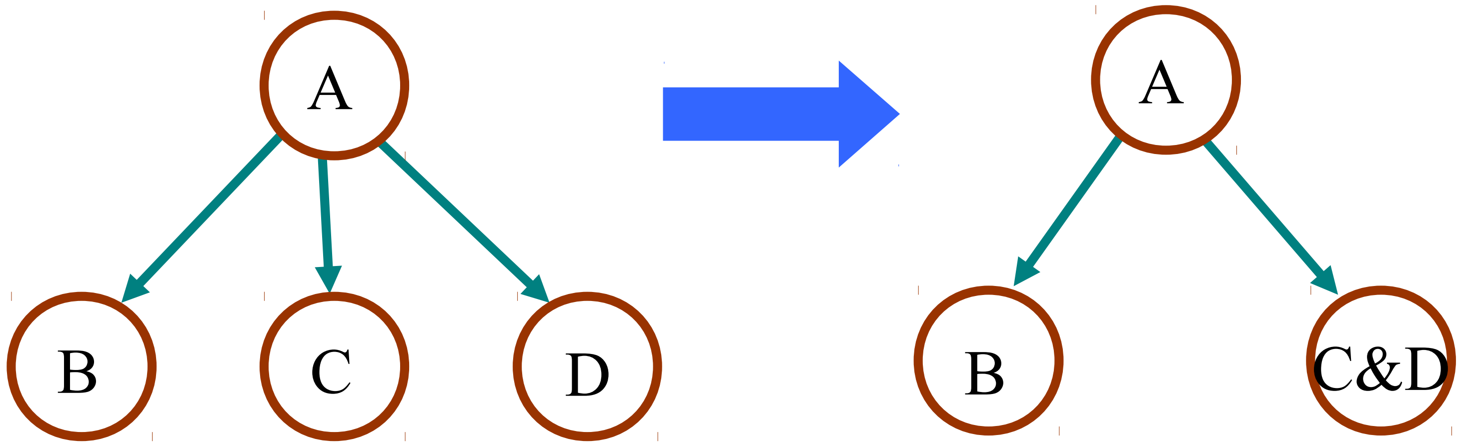
Problems with cutset conditioning 2

To combine the result of the many possible instantiations of the cutset we need to find priors each one.

There may not be enough data to do this reliably.

Joining dependent variables (node clustering)

If C and D are dependent (given A) we can combine them into one new variable by joining them.



Instantiating Joined Variables

If we instantiate one of C or D, but not the other, (eg D=d2) we must either re-calculate the link matrix:

$P(C\&D|A) \Rightarrow P(C|A)_{D=d2}$

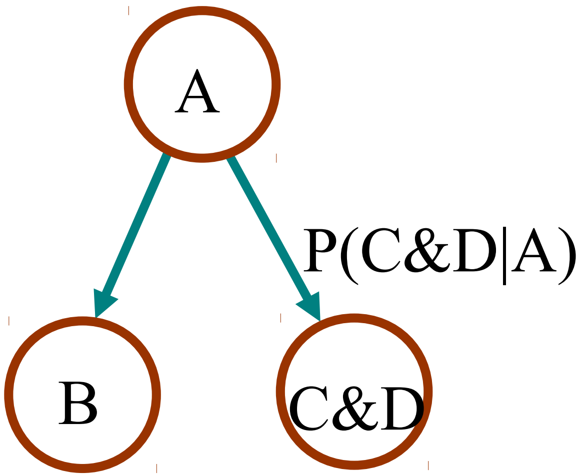
or instantiate several states:

Given C&D has states

$\{c1d1, c2d1, c3d1, c1d2, c2d2, c3d2\}$

If we instantiate D=d2 we have

$\lambda(C\&D) = \{0,0,0,1,1,1\}$



Limitation of joining variables

If two variables are to be joined, C having |C| states and D having |D| states the new variable C&D will have |C|*|D| states.

The increased number of states is undesirable and limits the applicability of this approach

In the limit a fully connected network degenerates into one node!

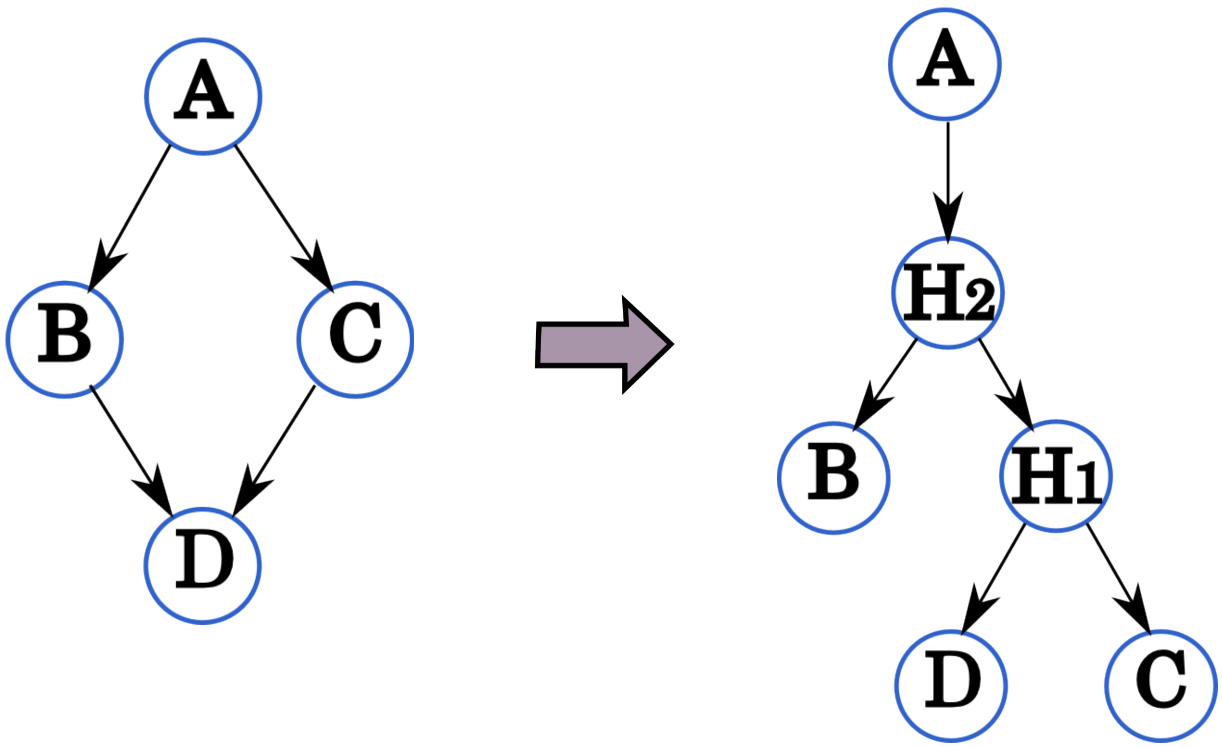
Limitation of joining variables

The number of states in joined variables goes up exponentially. This means that the method is again n-p hard.

The number of entries in the conditional probability matrices also goes up dramatically, meaning more data is needed to estimate them. As with cutset conditioning the method is limited by the size of the available data.

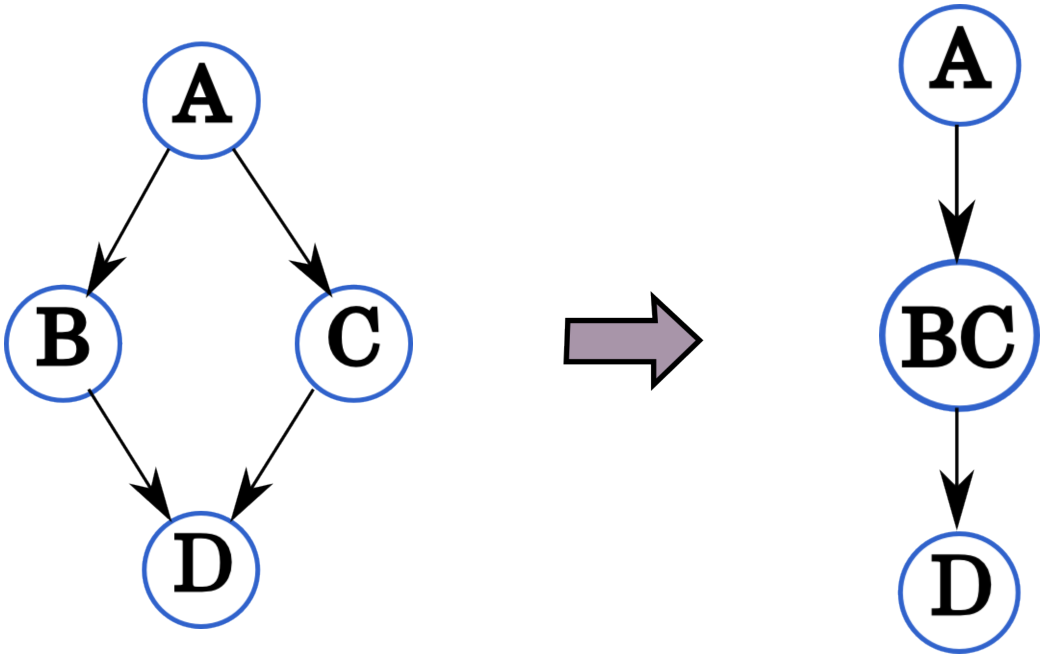
A Second Node Clustering example

In the last lecture we solved the loop problem by the introduction of hidden nodes



A Second Node Clustering example

Node clustering provides a simpler and more elegant solution:



However the method still scales badly especially if many nodes are involved in the cluster:

Join Trees (also called Junction trees)

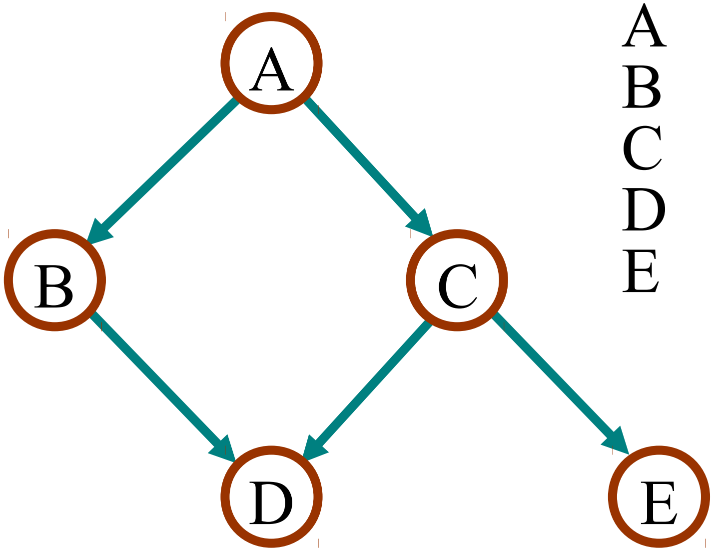
Join trees are a generalisation of the previous method.

Given a network, with all dependencies included as arcs, the objective is to find a systematic way of joining the variables such that propagation is possible between groups of variables.

This idea was originally pioneered by Lauritzen and Speigelhalter.

Example - Overview

We shall use the following medical example which is taken from Neapolitan's first book:

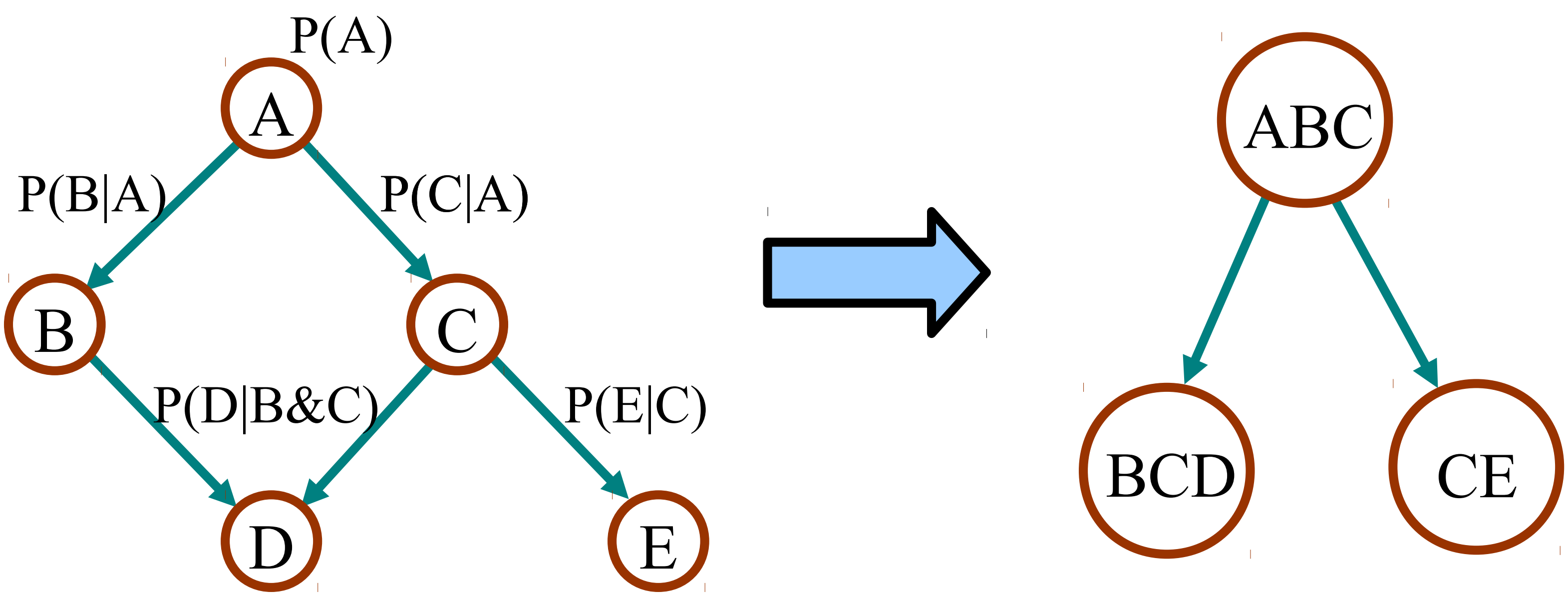


A	{a1,a2}	Metastatic Cancer
B	{b1,b2}	Total Serum Calcium Increase
C	{c1,c2}	Brain Tumour
D	{d1,d2}	Coma
E	{e1,e2}	Papilledema

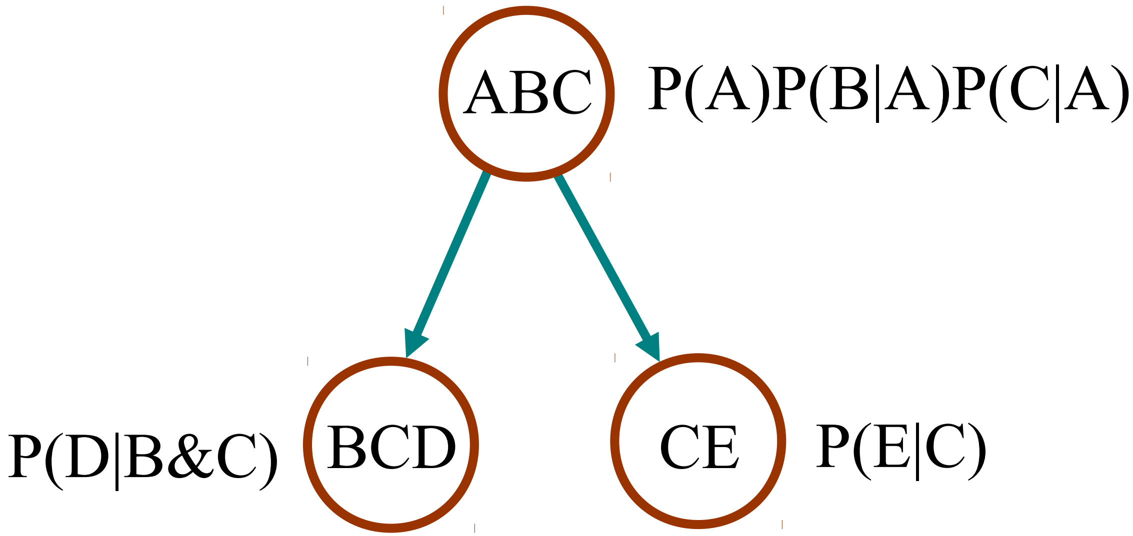
The join tree

We transform the original tree into a join tree.

The join tree has nodes grouped according to the connectivity in the original graph



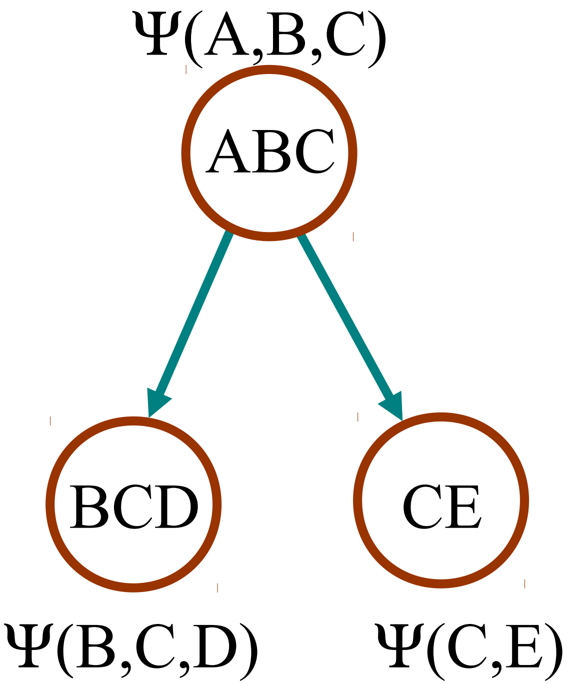
Join tree nodes



Each node in the join tree is give a potential table which is created from the original conditional probability matrices.

It contains evidence for the variables of that node.

Join tree potential tables

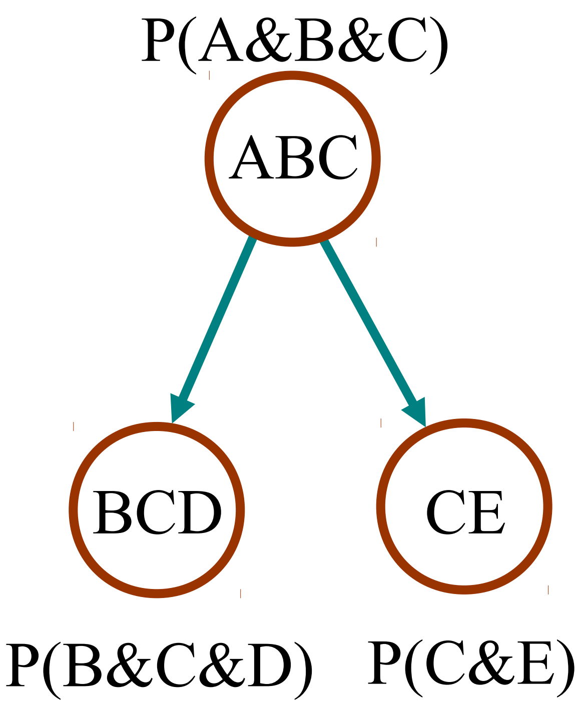


The potential table associated with each node has an entry for each joint state.

It could be a conditional probability table based on groups of nodes

$$\Psi(B,C,D) = P(D|B,C)$$

Join tree nodes - joint probabilities

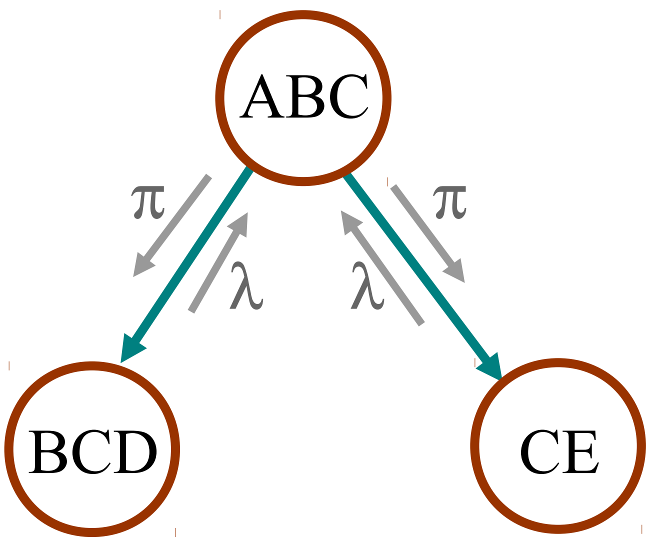


For inference, a calculation is made of the joint probabilities of each node. For example

$$P(B\&C\&D) = P(D|B\&C)\ P(B\&C)$$
$$= \Psi(B,C,D)\ P(B\&C)$$

The probabilities of individual variables can then be found by marginalisation

Instantiation and propagation



After an instantiation, computation is done by propagation in just two passes, up followed by down.

In the upward pass the potential functions are made into conditional probability tables. In the downward pass the joint probability tables are calculated.

Potential Representations

A potential representation is a collection of subsets Wi of the original variables V (nodes in the original tree). Each subset has a potential table (or function) ψ with the property:

$$P(V) = \prod_{i} \psi(Wi)$$

That is to say, the product of all the potential tables is the joint probability of the original set of variables.

Potential Representations

$$P(V) = \prod_i \psi(Wi)$$

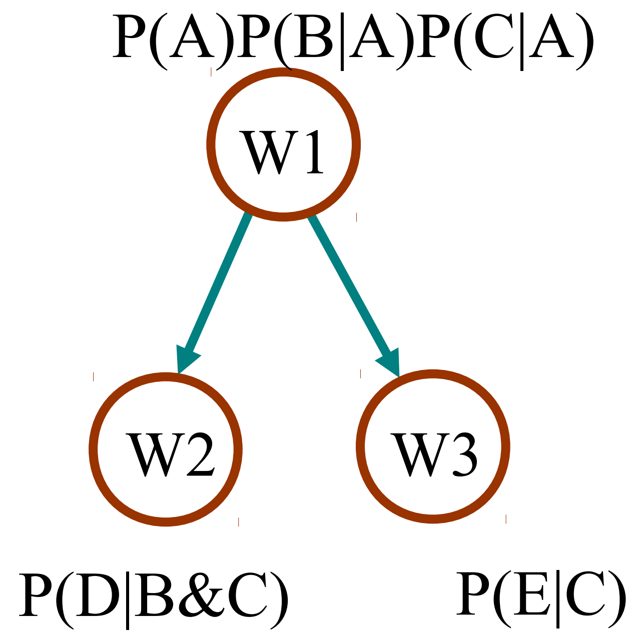
In our previous example:

$\psi(W1) = P(A)P(B|A)P(C|A)$

$\psi(W2) = P(D|B\&C)$

$\psi(W3) = P(E|C)$

$P(V) = P(A\&B\&C\&D\&E)$



$W1 = \{A,B,C\}$

$W2 = \{B,C,D\}$

$W3 = \{C,E\}$

Intersections

For a potential representation, given an ordering, we can define some intersection sets at each join tree node:

$$Si = Wi \cap (W1 \cup W2 \cup W3 \ldots\ldots \cup Wi\text{-}1)$$

(The variables in Wi which are in any parent set)

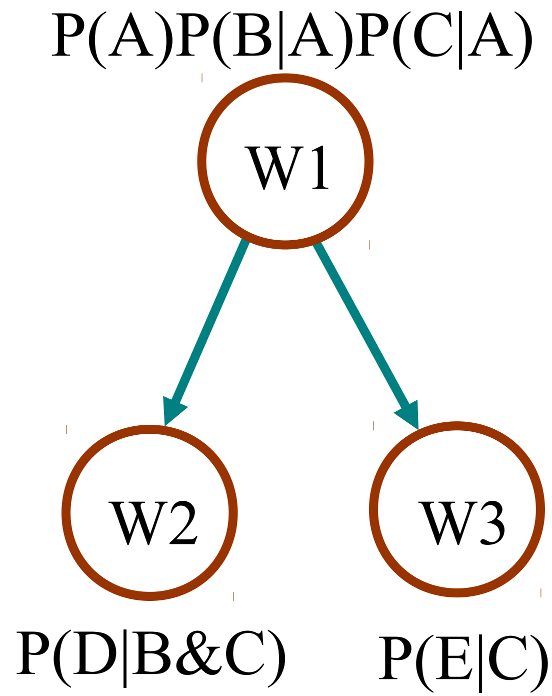
NB Lower index => parent

$$Ri = Wi - Si$$

(The variables in Wi that have not appeared higher in the tree)

Intersection Sets

	Wi	Si	Ri
1	A B C		A B C
2	D B C	B C	D
3	EC	C	E



W: The variables that belong to a node

S: The variables that appear higher up in the tree

R: The variables that have not been seen before

The running intersection property

To permit propagation, the potential representation must have the running intersection property:

Given an ordered set of subsets of our variables V, for any adjacent sets i and j such that j is a parent of i (j<i):

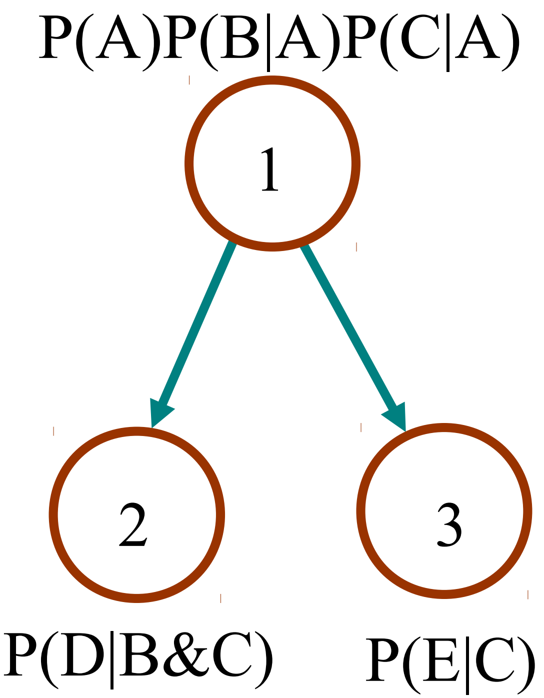
$$Si \subseteq W_j$$

If this is so we can write:

$$P(V) = P(W1) \prod_{i=2}^{p} P(Ri|Si)$$

Running Intersection Property

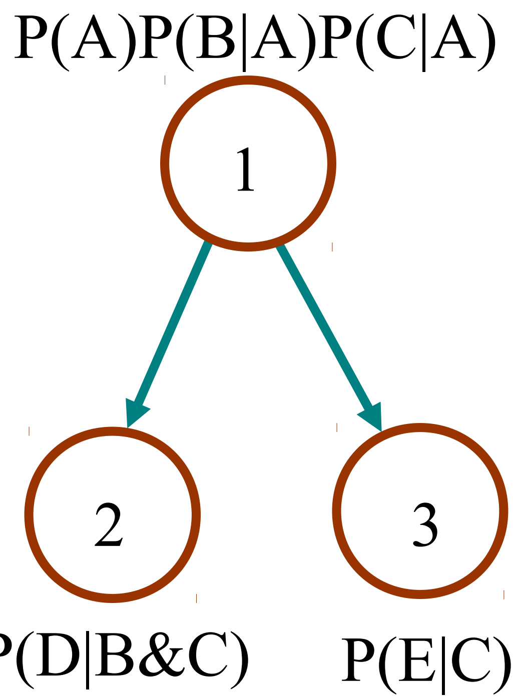
	Wi	Si	Ri
1	A B C		A B C
2	D B C	B C	D
3	EC	C	E



Everything in set S must also be in an immediate parent

Running Intersection Property

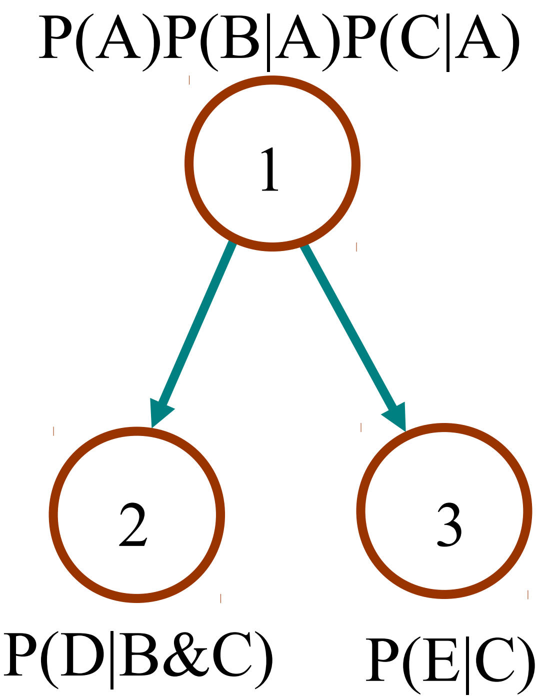
	Wi	Si	Ri
1	A B C		A B C
2	D B C	B C	D
3	EC	C	E



Everything in set S must also be in an immediate parent

Running Intersection Property

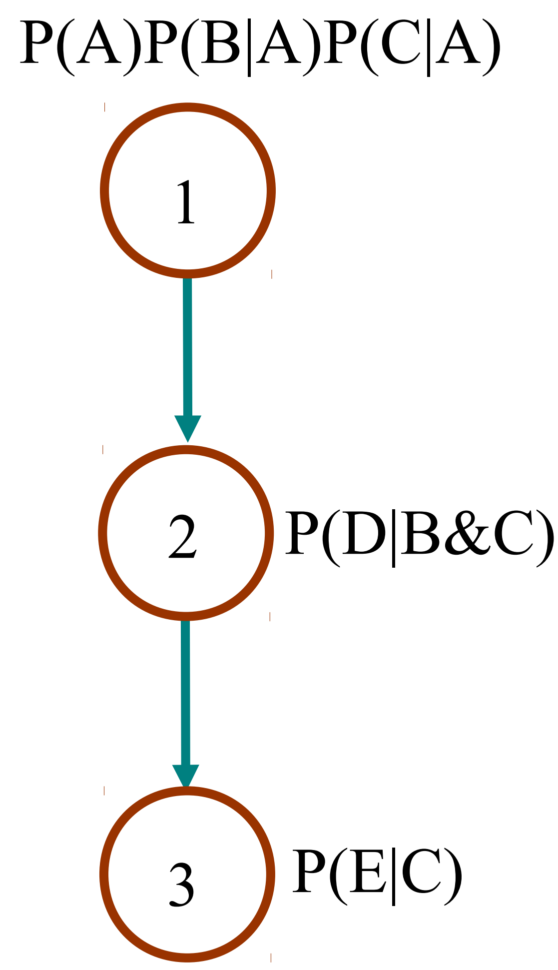
	Wi	Si	Ri
1	A B C		A B C
2	D B C	B C	D
3	EC	C	E



Everything in set S must also be in an immediate parent

Running Intersection Property - another tree

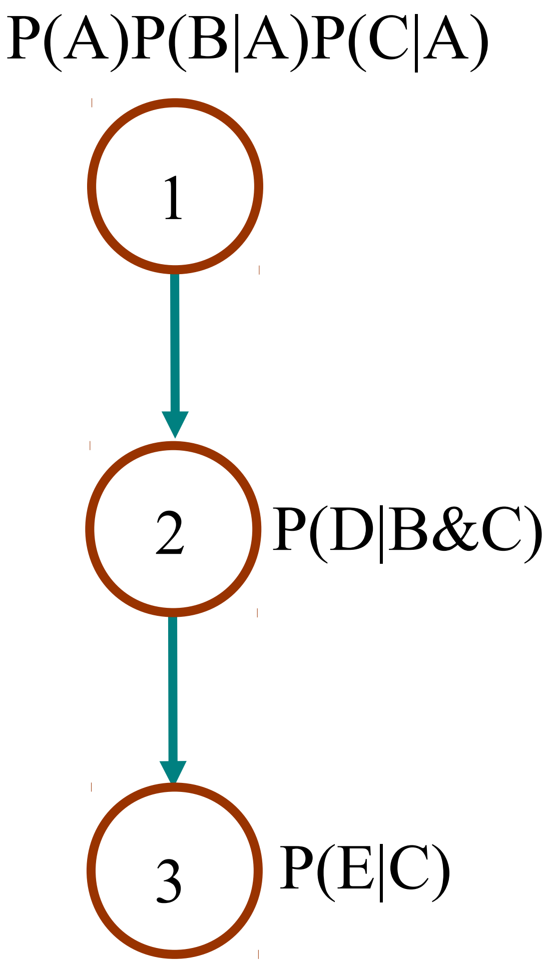
	Wi	Si	Ri
1	A B C		A B C
2	D B C	B C	D
3	EC	C	E



Everything in set S must also be in an immediate parent

Running Intersection Property - another tree

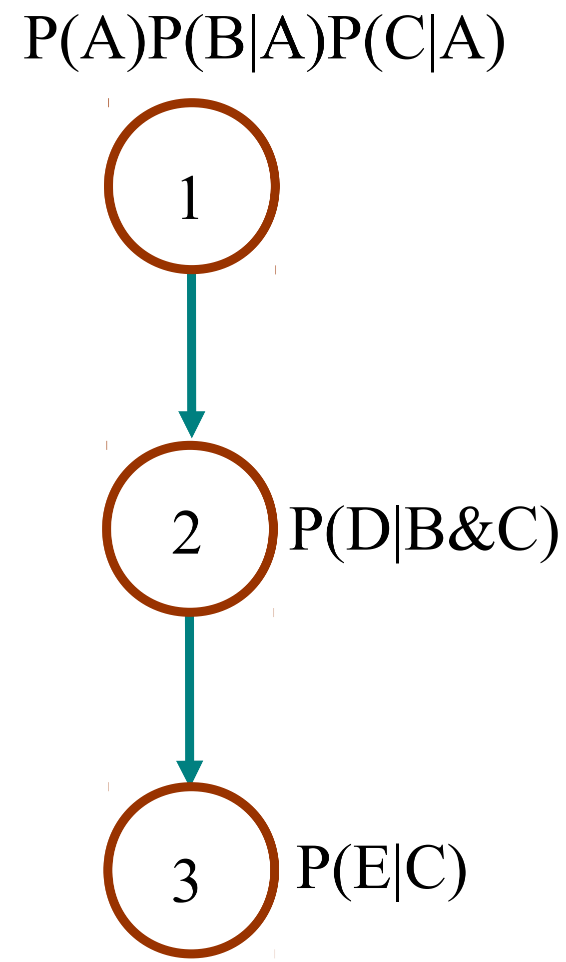
	Wi	Si	Ri
1	A B C		A B C
2	D B C	B C	D
3	EC	C	E



Everything in set S must also be in an immediate parent

Running Intersection Property - another tree

	Wi	Si	Ri
1	A B C		A B C
2	D B C	B C	D
3	EC	C	E



Everything in set S must also be in an immediate parent

Summary

The ordering is set up so that each node of the join tree has an associated set of variables $R \cup S$

R is a set of variables that have not been seen above

S is a set of variables which must appear in the immediate parents.

The running intersection property ensures we will have a link matrix $P(R|S)$ for each node.

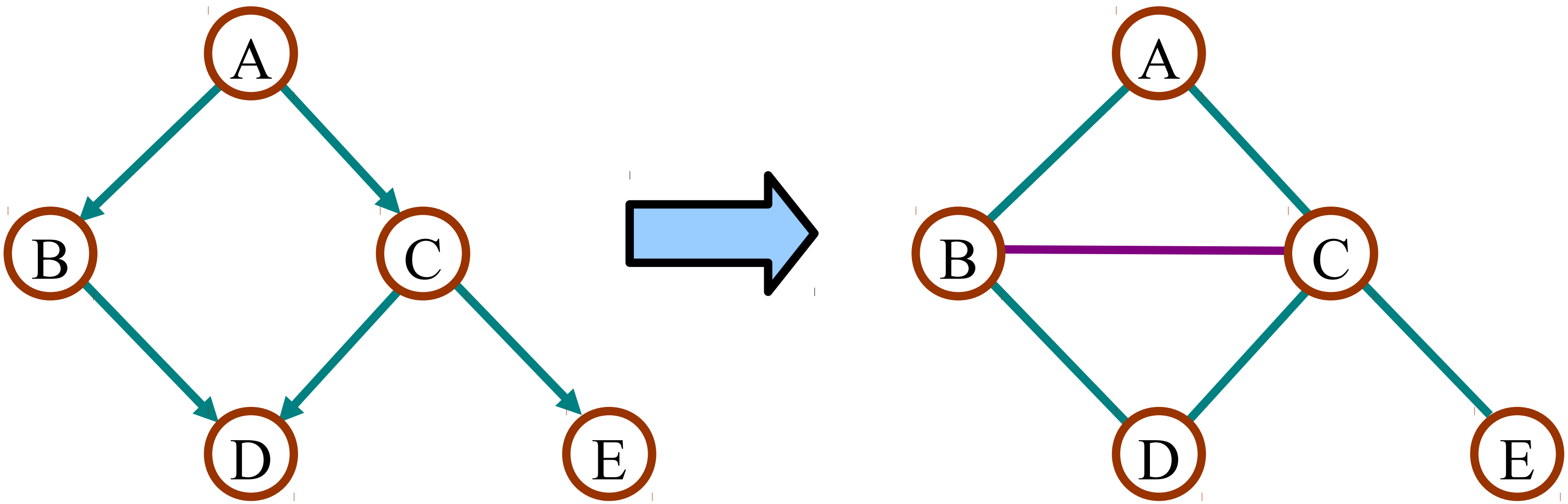
Finding the ordered subset of the nodes

An ordered subset of the variables can always be found from a given causal graph as follows:

1. Moralise the graph (join any unjoined parents)
2. Triangulate the graph
3. Identify the cliques of the resulting graph
4. Find an ordering with the running intersection property
5. For each clique Cli find the set of its variables {Xi} whose parents Pa(Xi) also belong to the clique. Iinitialise the clique potential function y as follows:

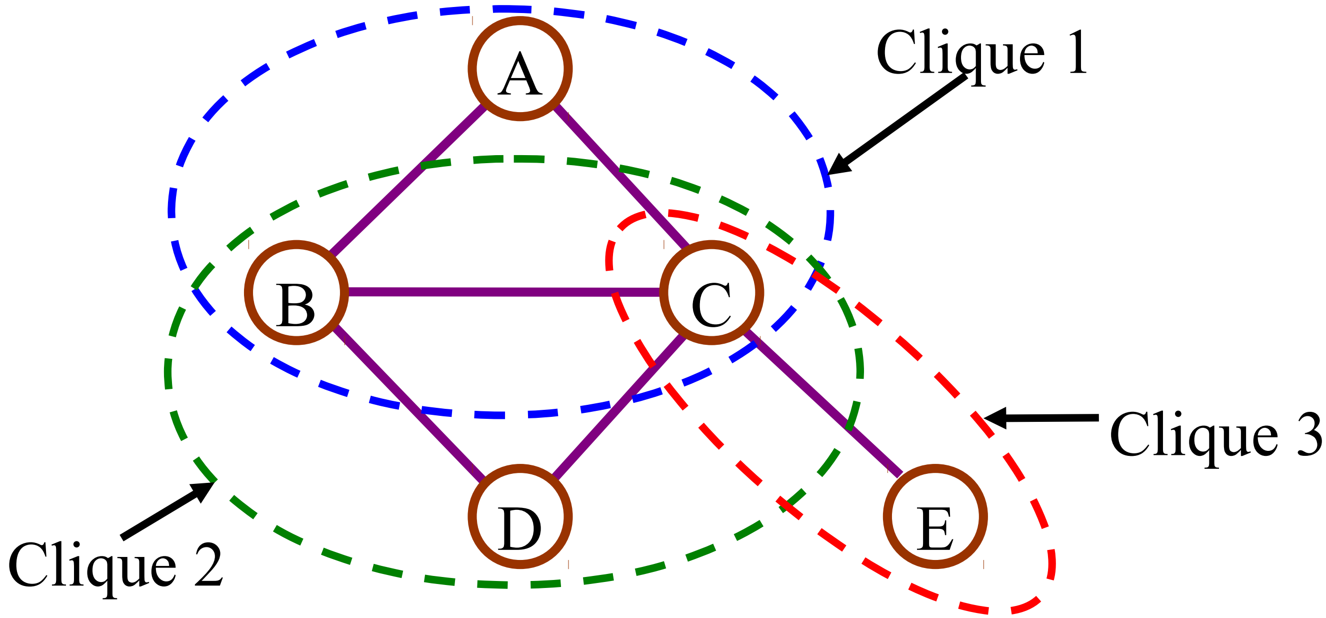
$$\psi(Cli) = \psi(Wi) = \prod_{\{Xi\}} P(X|Pa(X))$$

1,2. Moralising and triangulation



After moralising this graph is triangulated so there is no need for a triangulation step

3. Finding the Cliques



A clique is a maximal set of nodes in which every node is connected to every other

4. Ordering the Cliques

Find an ordering of the cliques that has the running intersection property. This can be done in many ways. The simplest is a search algorithm.

- Start with a clique containing a root of the original network.
- Find all nodes that can be its children – ie those for which any variable appearing higher up in the evolving tree is in the parent.
- Recursively search from the children till all cliques are joined.

5. Initialising Potential functions

For each clique we find the set of variables whose parents are also in the same clique. These will be the variables in the R set for the clique

Each variable of the original graph will appear in only one R set. The moralisation step ensures that this is the case.

5. Initialising Potential functions

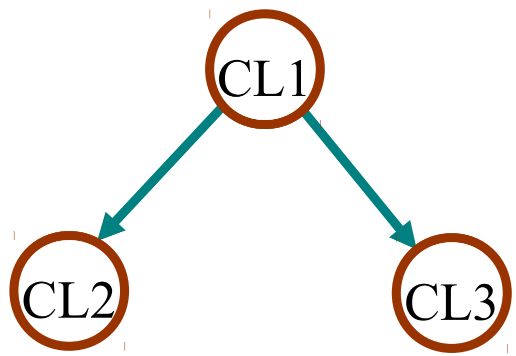
The potential functions can then be found for each clique. They are the product of the conditional or prior probability tables with distributions over the variables in the R set:

Index	W	S	R	Ψ
1	ABC		ABC	$P(A)P(B\|A)P(C\|A)$
2	BCD	BC	D	$P(D\|B\&C)$
3	CE	C	E	$P(E\|C)$

The Join tree

The join tree is now completed.

In all future probability calculations we use just the join tree, disregarding our original network



The two possible join trees