Bayesian Networks

© Jorge Salvador Marques, 2002

Challenge

Sherlock Holmes was asked to solve the following problem:

A man was murdered last night and the Police has three suspects. A knife was found close to the body and the tests showed that it had the fingerprints of suspect 3. A neighbor saw a man running out of the house in which the murder occurred and the man had the same height of suspect 1 being much smaller than suspects 2 and 3.

Which suspect is the murderer with highest probability ?

© Jorge Salvador Marques, 2002

Summary

- Motivation
- Bayesian nets
- Inference in Bayesian nets
- Pearl algorithm
- Frey algorithm for factor graphs
- Junction tree

© Jorge Salvador Marques, 2002

Roots

Graphical models were proposed in the 1st half of the 20th century in several fields e.g., in genetics and later in AI.

Recently they began to attract the attention of the electric engineering community as well as the attention of statisticians.

Mile stones:

- Wright (1921) geneticist – proposed a graphical representations for probabilities (severely criticized by statisticians).
- Howard e Matheson (1981) – developed influence diagrams for decision analysis.
- J. Pearl (1982) proposed an algorithm for the propagation of beliefs in trees as a way to model human reasoning. Later he extended this algorithm to Bayesian networks without multiple paths.

© Jorge Salvador Marques, 2002

Why do we need Bayesian networks ?

Consistent reasoning should obey the rules of probabilistic calculus (Polya).

Key question: *How does the human brain perform inference with large number of variables ?*

Paradox (Pearl,97)

Consider a collection of random variables $x_1, ..., x_n$.

Classic probability theory suggest that these variables should be characterized by a joint distribution $P(x_1, ..., x_n)$.

But it is almost impossible to learn joint distributions for a large number of variables without making strong assumptions.

© Jorge Salvador Marques, 2002

Why do we need Bayesian networks ?

The inference of x_i from x_j requires the computation of marginal probabilities

$$p(x_i \mid x_j) = \frac{\sum_{x:\text{known } x_i, x_j} p(x_1,\ldots,x_n)}{\sum_{x:\text{known } x_j} p(x_1,\ldots,x_n)}$$

which can not be computed in most problems.

Human reasoning does not work in this way ! It is easier to evaluate a proposition based on conditional probabilities $p(x_i|x_j)$ than on the joint distribution.

Assumptions (Pearl):

- reasoning should be based on marginal and conditional probabilities;
- the computational efficiency is achieved by an efficient representation of independence among different sets of variables.

© Jorge Salvador Marques, 2002

Key Issues

How to represent a set of random variables using conditional probabilities ?

How to represent independence among subsets of variables ?

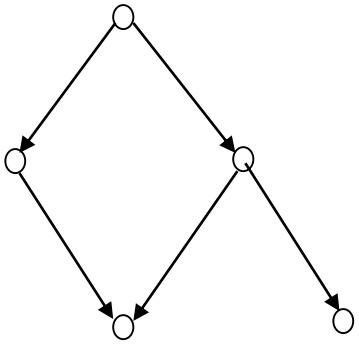
Graphical models provide answers to these questions namely

- **Bayesian networks** based on directed acyclic graphs (DAGs)
- **Markov random fields** based on undirected graphs.

© Jorge Salvador Marques, 2002

DAG

A direct acyclic graph (DAG) consists of a set of nodes and directed edges between nodes,



Each node may have several parents (causes) and several children (efects).

No cycles are allowed if we follow the edge directions.

Links mean causal direct dependencies.

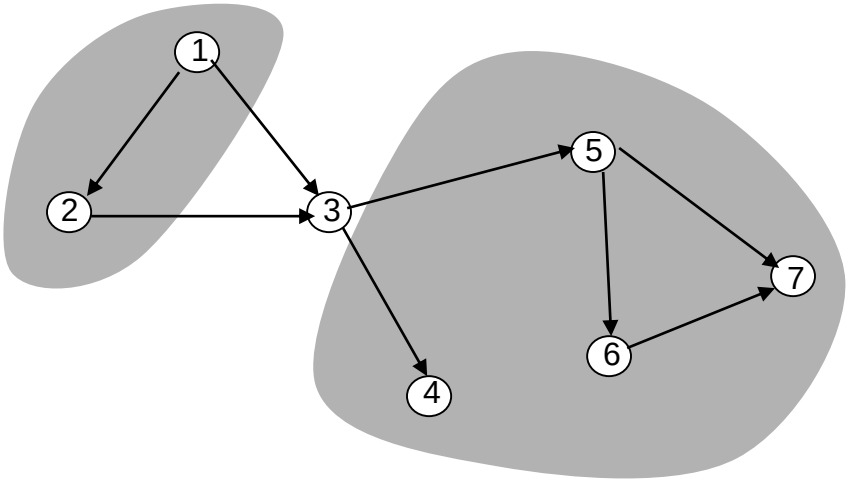
© Jorge Salvador Marques, 2002

Ancestral Order

In a DAG it is possible to define a *past* and a *future* for each node.

Proposition: It is always possible to order the nodes of a DAG in such a way that the descendents of each node have a higher index value.

non descendents of node 3 ($\neq$3)



descendents of node 3

This property is violated if the graph contains cycles !

© Jorge Salvador Marques, 2002

Bayesian Net

Given a set of random variables $x=(x_1, ..., x_n)$, a Bayesian net (BN) is a probabilistic model for x defined by

- *a DAG, each node being associated to a random variable* x_i
- *conditional probability functions associated to each node:* $P(x_i|a_i)$ *where* a_i *denotes the parents of* x_i.

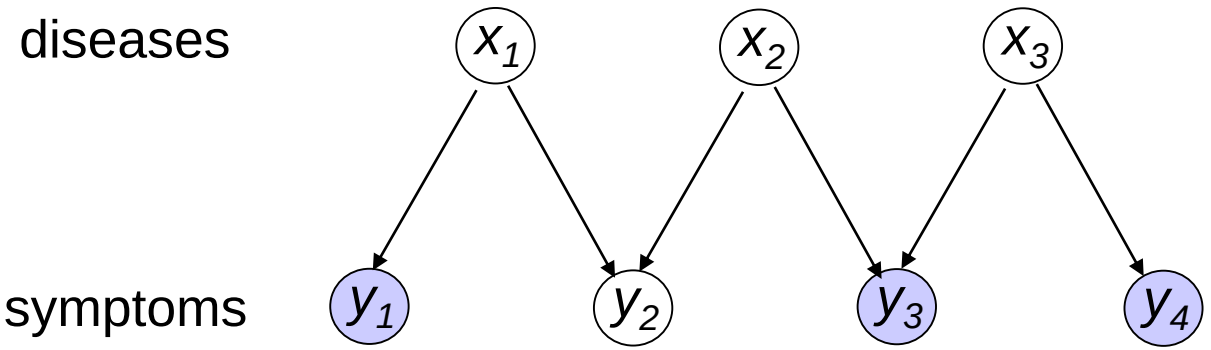
Furthermore it is assumed that

$$P(x_i|nd_i) = P(x_i|a_i) \qquad \textit{Markov property}$$

where nd_i are the non descendent nodes of x_i. Therefore, the parents summarize all the information contained in the non descendant nodes of x_i.

© Jorge Salvador Marques, 2002

Example

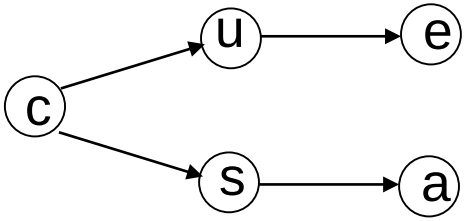


Note: observed nodes are painted in gray

© Jorge Salvador Marques, 2002

Sherlock Holmes

The problem is addressed using five random variables



c – criminal
u – the last person who handled the knife
e – expert exam
s – person who left the scene
a - height

Characterization:

$$p_C = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

c

p(u/c)	1	2	3
1	.9	.05	.05
2	.05	.9	.05
3	.05	.05	.9

u

P(e/u)	1	2	3
1	.8	.1	.1
2	.1	.8	.1
3	.1	.1	.8

c

P(s/c)	1	2	3
1	.8	.1	.1
2	.1	.8	.1
3	.1	.1	.8

s

P(a/s)	1	2	3
1	.6	.3	.1
2	.1	.6	.3
3	.1	.4	.5

© Jorge Salvador Marques, 2002

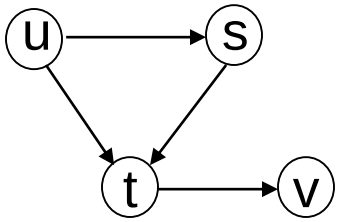
Joint Distribution

Property: if $x=(x_1, ..., x_n)$ is described by a Bayesian network

$$P(x) = \prod_i P(x_i \mid a_i)$$

Bayesian networks factorize the joint distribution !

Example



P(u,s,t,v) = P(v|t) P(t|u,s) P(s|u) P(u)

© Jorge Salvador Marques, 2002

Proof

Suppose the nodes are ordered according to an ancestral order (descendents have a higher index).

Then

$$p(x_1,\ldots,x_n) = \prod_i p(x_i \mid \underbrace{x_1,\ldots,x_{i-1}}_{\text{non descendents}}) \qquad \text{(def. of conditional prob.)}$$

$$= \prod_i p(x_i \mid a_i) \qquad \text{(Markov property)}$$

© Jorge Salvador Marques, 2002

Independence

The efficiency the inference operations depends on the ability to efficiently represent the independence of random variables.

Two key questions are:

- do Bayesian networks represent independence ?
- is it possible to determine if two subsets of variables are independent by the inspection of the graph ?

Difficulty: independence depends on the graph and on the conditional distributions associated to the nodes.

© Jorge Salvador Marques, 2002

d - Separation

Definition: Let x^A, x^B, x^S, be three disjoint sets of variables of a BN. x^A is denoted **d-separated** from x^B given x^S iif

$$P(x^A / x^B, x^S) = P(x^A / x^S) \quad , \qquad \text{for all } P$$

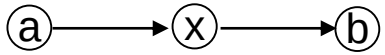
The independence of variables depends both on the graph and on the probability distribution P.

d- separation is stronger since it depends only on the graph topology.

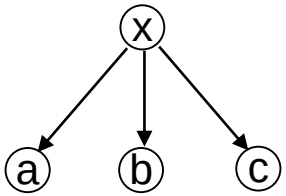
© Jorge Salvador Marques, 2002

Evidence Propagation

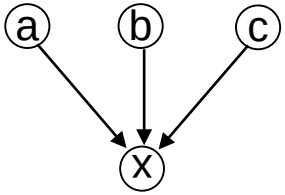
serial connection



diverging connection



converging connection



Evidence may be transmitted in a serial or diverging connection if x is not instantiated.

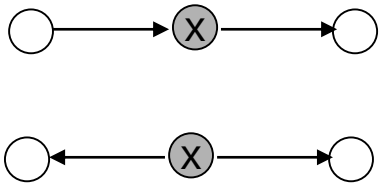
Evidence may only be transmitted in a converging connection if x or one of its descendand is instantiated.

© Jorge Salvador Marques, 2002

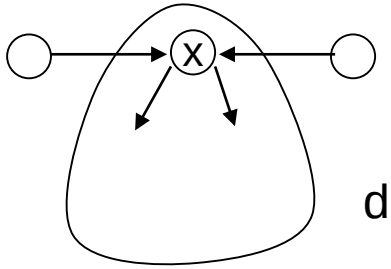
D-Separation Rule

x^A, x^B are d-separated by x^S if all the paths from x^A to x^B are blocked. A path is blocked if there is

- a non convergent connection with a variable $x \in x^S$ or
- a convergent conection such that x and its descendants do not belong to x^S



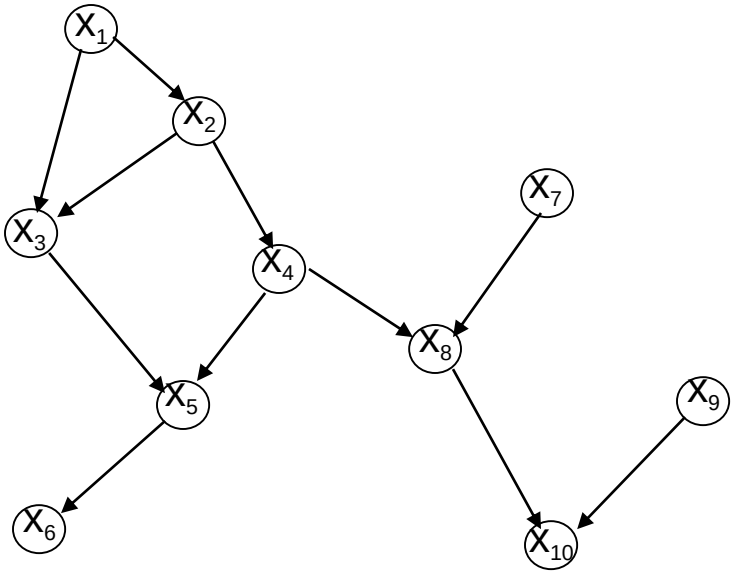
x belongs to x^S



do not belong to x^S

© Jorge Salvador Marques, 2002

Example



x_1 e x_4 are d-separated by x_2

x_1 e x_4 are not d-separated given $\{x_2,x_5\}$
x_5 creates dependency

x_4 e x_9 are d-separated

x_4 e x_9 are not d-separated given x_{10}
x_9 creates dependency
in its ancestors

© Jorge Salvador Marques, 2002

Suzan Family

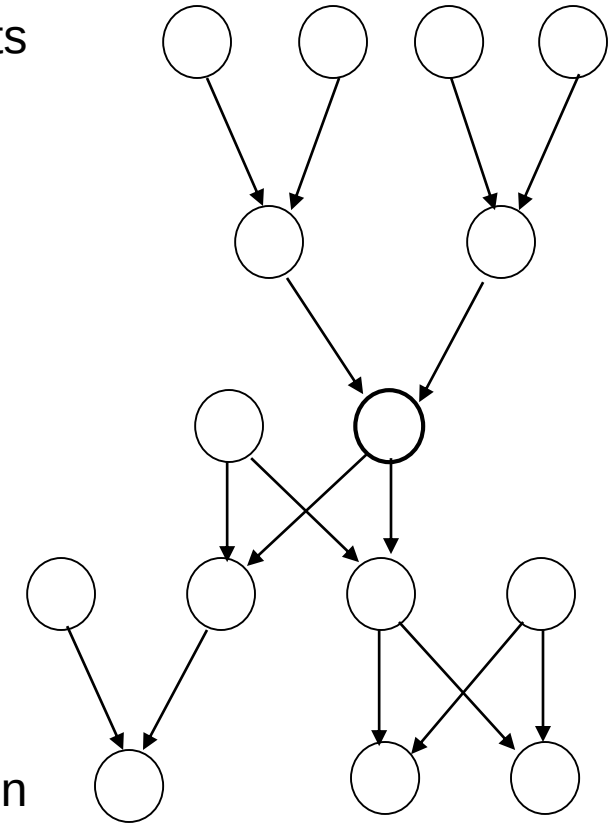
grand parents

parents

Susan

children

grand children



links represent genetic dependency

It is assumed that the genetic description of Susan parents is known

Questions: is the prediction of Susan genetic code modified if we know the genes of the grand parents, sons, grand children, or by the parents of the grand children which are not sons ?

© Jorge Salvador Marques, 2002

Inference Methods

Inference aims to compute the distribution of a set of hidden variables given the observations, y. The simplest case is to compute $P(x_i/y)$ (belief) where x_i is a hidden variable.

There are several algorithms to compute the beliefs $P(x_i/y)$.

Inference in singly connected nets

- message passing (Pearl)
- multiply-add (Frey)

Inference in multiply connected nets

- junction tree (Jensen)
- Monte Carlo methods
- variational methods

© Jorge Salvador Marques, 2002

Pearl Method

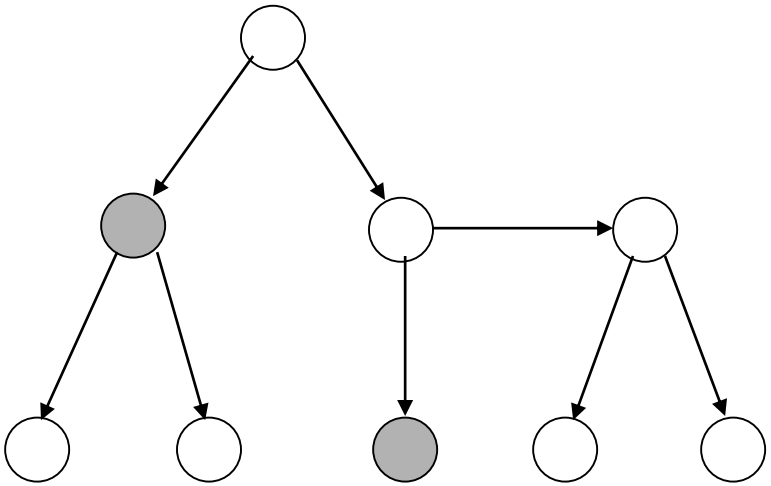
Pearl method is one of the first inference methods proposed in the literature. It extends the forward-backward algorithm proposed by Baum for HMM.

The algorithm was proposed by Pearl in 1982 for BN with tree topology and it was extended in 1983 to general BN without loops.

© Jorge Salvador Marques, 2002

Trees

Tree networks are BN such that each node has a single cause (parent).



observed (evidence)

hidden

Inference goal: compute the distribution of each variable given evidence.

© Jorge Salvador Marques, 2002

Evidence Decomposition

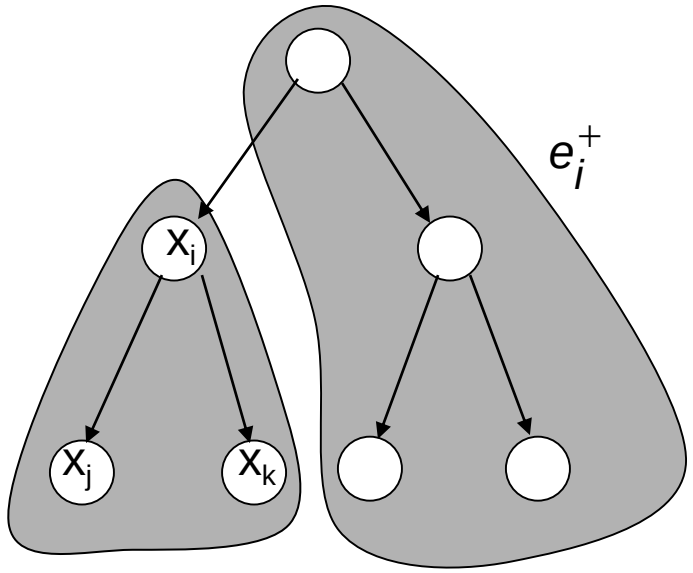
Each node splits the evidence in two disjoint parts (present-past and future)

e_i^- descendents of the i-th node

e_i^+ other (non descendent) nodes

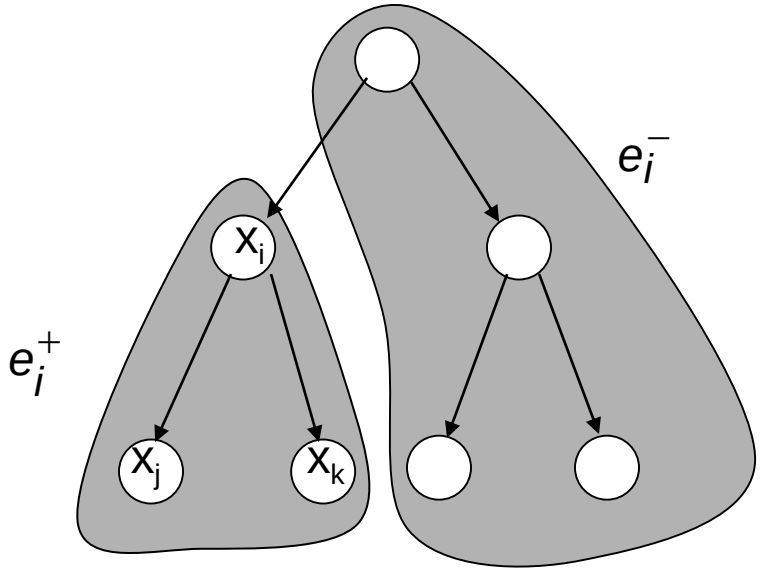
e_i^- Can also be decomposed as follows

$$e_i^- = e_j^- \cup e_k^-$$



© Jorge Salvador Marques, 2002

Auxiliar Probabilities



$$\alpha(x_i) = P(x_i / e_i^+)$$

$$\beta(x_i) = P(e_i^- / x_i)$$

β_i are obtained by a bottom-up recursion.

α_i are obtained by a top-down recursion using β_i variables.

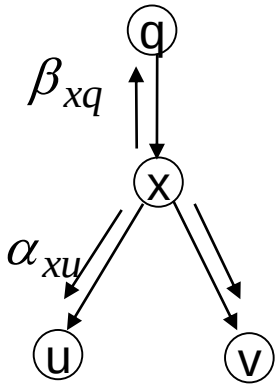
Inference: $P(x_i \mid y) = c\alpha(x_i)\beta(x_i)$ c – normalization factor

© Jorge Salvador Marques, 2002

Pearl Algorithm for Trees

Proposition: the node conditional probabilities, given the evidence e, can be obtained as follows:

$$p(x \mid e) = c\alpha(x)\beta(x)$$



where $\alpha(x) = \sum_q P(x \mid q)\,\alpha_{qx}(q)$, $\beta(x) = \beta_{ux}(x)\beta_{vx}(x)$

are obtained by a message passing procedure:

Bottom-up messages

$$\beta_{xq}(q) = \sum_x P(x \mid q)\beta(x)$$

Top-down messages

$$\alpha_{xu}(x) = c\alpha(x) \prod_{\text{other siblings v}} \beta_{vx}(x)$$

Special cases:

if *x* is the root $\alpha(x)=P(x)$; if *x'* is an observation of *x*, $\beta(x)=\delta(x,x')$, if *x* is a hidden leave $\beta(x)=1$

© Jorge Salvador Marques, 2002

Proof

i)

$$P(x \mid e) = c' P(e_X^-, e_X^+, x) = c' P(e_X^- \mid e_X^+, x) P(e_X^+, x)$$

$$= cP(e_X^- \mid e_X^+, x) P(x \mid e_X^-) = c\alpha(x)\beta(x)$$

ii) *x* observed

$$\beta(x) = P(e_X^- \mid x) = \delta(x, x')$$

$$Def: \alpha(x) = P(x \mid e_X^+)$$

$$\beta(x) = P(e_X^- \mid x)$$

x non observed

$$\beta(x) = P(e_X^- \mid x) = P(e_U^-, e_V^- \mid x) = P(e_U^- \mid x) P(e_V^- \mid x)$$

Defining $\beta_{UX}(x) = P(e_U^- \mid x), \beta_{VX}(x) = P(e_V^- \mid x)$

$$\beta(x) = \beta_{UX}(x) \beta_{VX}(x)$$

The product has one factor for each son of node x.

© Jorge Salvador Marques, 2002

Proof (2)

iii)
$$\beta_{xq}(q) = P(e_x^- \mid q) = \sum_x P(e_x^-, x \mid q) = \sum_x P(e_x^- \mid x,q)P(x \mid q)$$
$$= \sum_x P(x \mid q)\beta(x)$$

iv)
$$\alpha(x) = P(x \mid e_x^+) = \sum_q P(x,q \mid e_x^+) = \sum_q P(x \mid q)P(q \mid e_x^+)$$

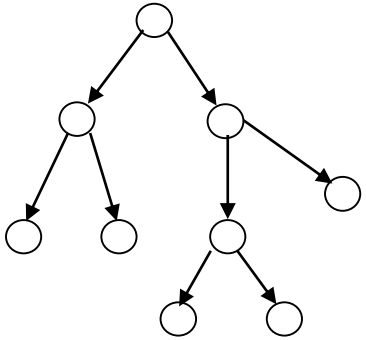
Definindo $\alpha_{qx}(q) = P(q \mid e_x^-)$ vem $\alpha(x) = \sum_q P(x \mid q)\alpha_{qx}(q)$

v)
$$\alpha_{xu}(x) = P(x \mid e_u^+) = P(x \mid e_v^-, e_x^+) = cP(e_v^- \mid x, e_x^+)P(x \mid e_x^+)$$
$$= c\beta_{vx}(x)\alpha(x)$$

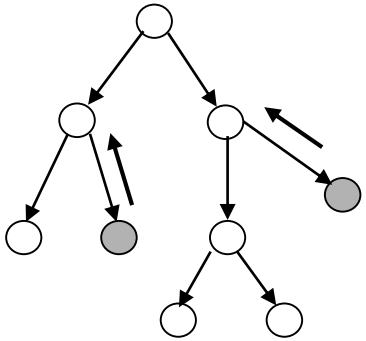
É neste passo que se cria a dependência do ciclo descendente em relação aos resultados do ciclo ascendente que têm que ser previamente calculados.

© Jorge Salvador Marques, 2002

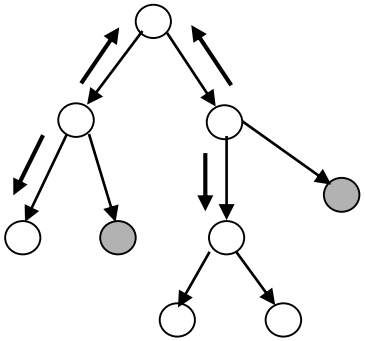
Mensage Propagation

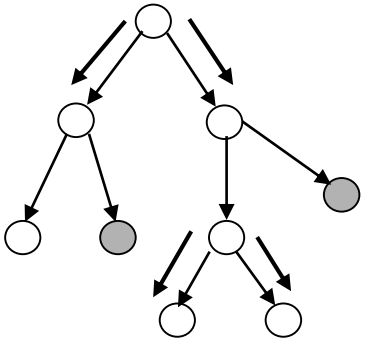


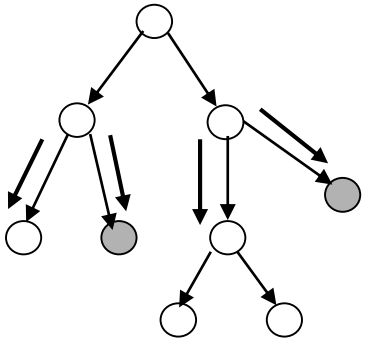
starting point

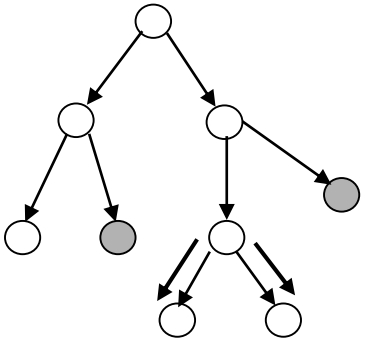


new data



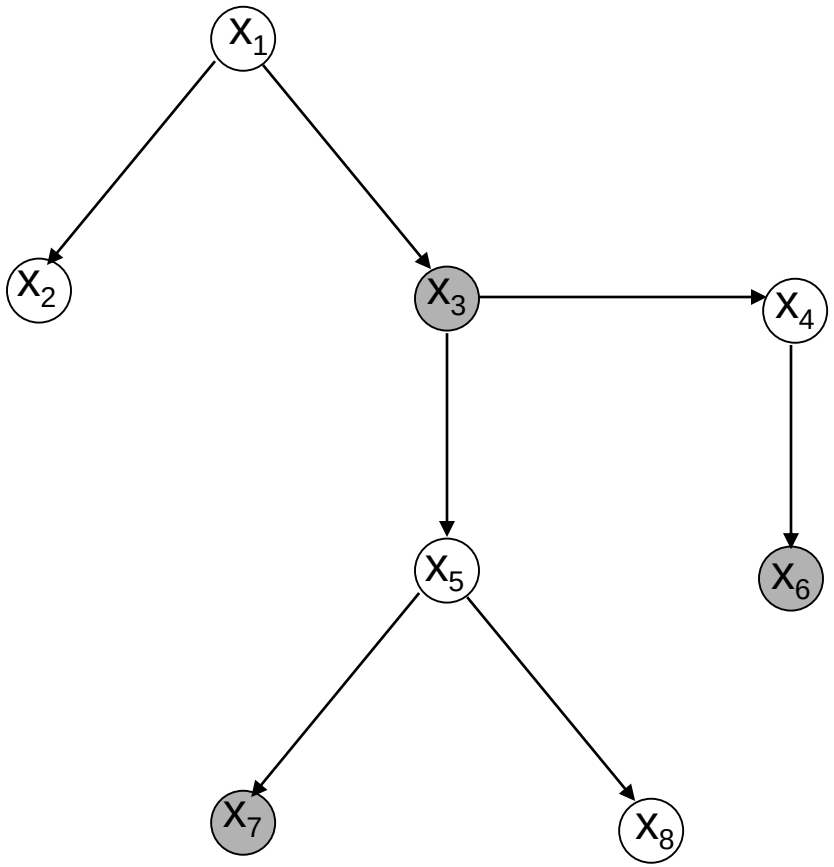






© Jorge Salvador Marques, 2002

Example



observations: $y_3=1$, $y_6=1$, $y_7=0$

Rot: $p_{x_1} = \begin{bmatrix} .7 \\ .3 \end{bmatrix}$

Transitions 12, 13, 46

father \ son	0	1
0	.9	.1
1	.5	.5

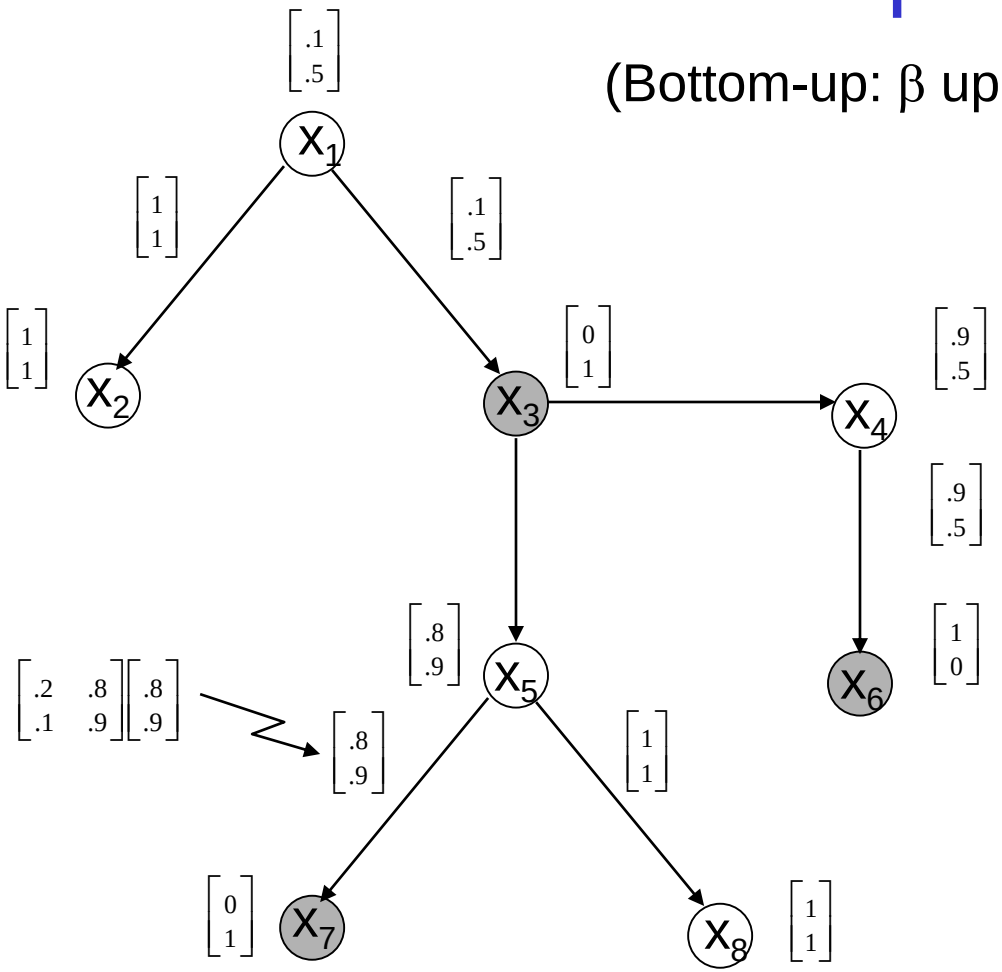
Other transitions

father \ son	0	1
0	.2	.8
1	.1	.9

© Jorge Salvador Marques, 2002

Example

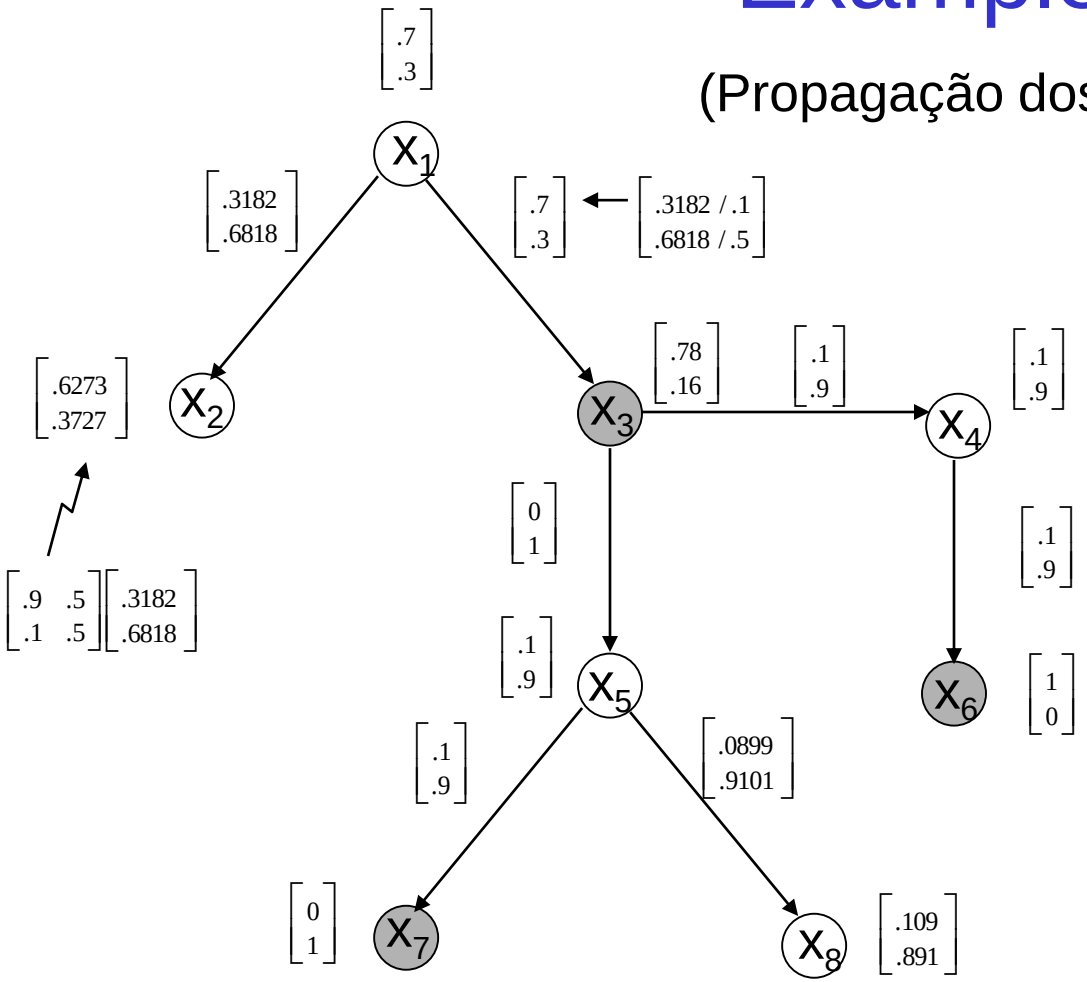
(Bottom-up: β update)



© Jorge Salvador Marques, 2002

Example

(Propagação dos αs)

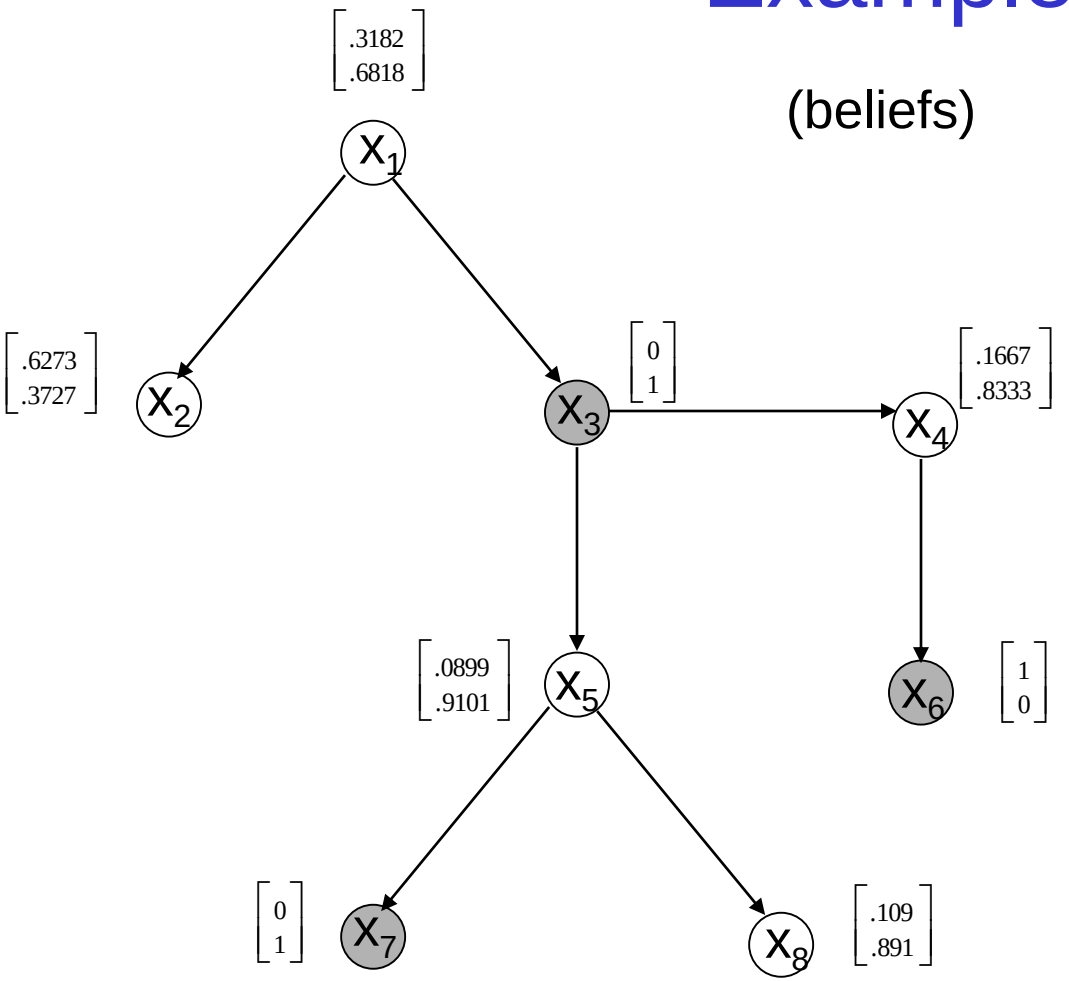


Note: beliefs $P(x_i/y)$ are not shown although they are recursively computed

© Jorge Salvador Marques, 2002

Example

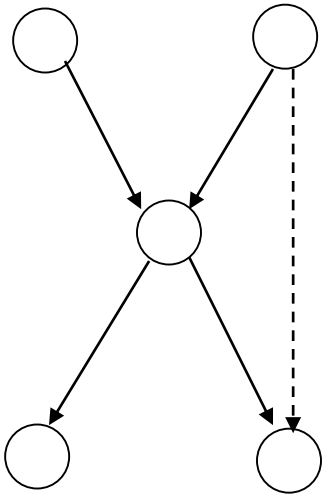
(beliefs)



© Jorge Salvador Marques, 2002

Singly Connected Networks

These networks allow multiple causes (fathers) for each node but not multiple paths between pairs of nodes.



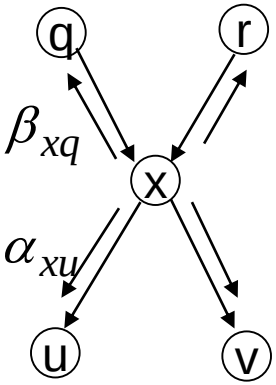
not allowed

Pear algorithm was generalized by Kim, Pearl, 1983 to deal with simply connected networks.

© Jorge Salvador Marques, 2002

Kim & Pearl Algorithm (1983)

(singly connected BN)



Update of internal variables

$$\alpha(x) = \sum_{q,r} P(x \mid q,r)\, \alpha_{qx}(q)\alpha_{rx}(r) \qquad \beta(x) = \beta_{ux}(x)\beta_{vx}(x)$$

$$p(x \mid e) = c\alpha(x)\beta(x)$$

Bottom up messages

$$\beta_{xq}(q) = c\sum_{x} \beta(x)\sum_{r} P(x \mid q,r)\alpha_{rx}(r)$$

Top down messages

$$\alpha_{xu}(x) = c\,\frac{P(x \mid y)}{\beta_{ux}(x)}$$

Special cases

If *x* is a root $\alpha(x)=P(x)$; if x' is an observation of *x*, $\beta(x)=d(x,x')$, if *x* is a hidden leave $\beta(x)=1$

© Jorge Salvador Marques, 2002

Factor Graphs

Factor graphs (FGs) reresent the decomposition of a joint distribution as a product of functions with less variables

They are used to represent

$$P(x) = \alpha \prod_{j=1}^{N} f_j(x^j)$$

x^j is the set of variables of f_j

Factor graphs have two types of nodes: nodes associated to random variables x_i and nodes associated to functions f_j. The f_j node is linked to the nodes of the variables in x^j .

Note: Frey claims that factor graphs are more general than Bayesian networks and Markov random fields.

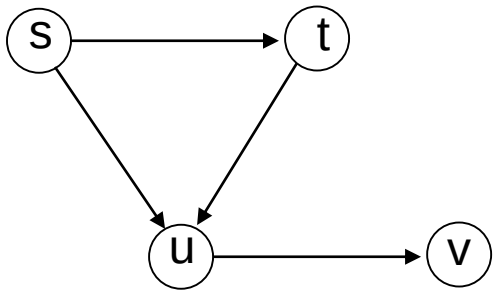
© Jorge Salvador Marques, 2002

From BN to FG

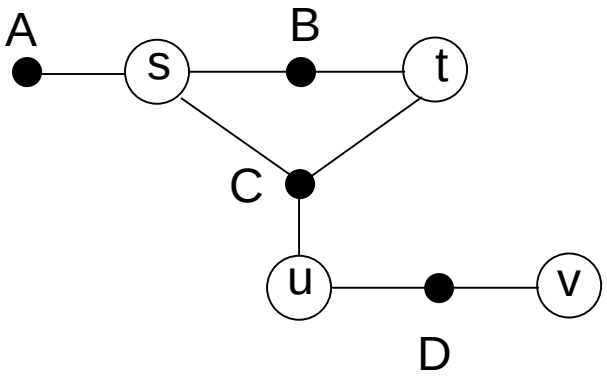
Any Bayesian network can be converted into a factor graph. We only have to creat additional nodes associated to the conditional distributions.

Example

Bayesian Network



Factor Graph



$x=\{s,t,u,v\}$

$P(x)=P(s)\ P(t|s)\ P(u|s,t)\ P(v|u)$

$f_A=P(s)$ $f_B=P(t|s)$ $f_C=P(u|t,s)$
$f_D=P(v|u)$

© Jorge Salvador Marques, 2002

Inference

The computation of $P(x_i/y)$ in a *singly connected* factor graph can be obtained by the **forward-backward** algorithm or by the **product-sum** algorithm.

Both are based on the same type of update rules based on message transmission between nodes. The difference lies only in the way messages are sent (bottom-up/top-down or asynchronous).

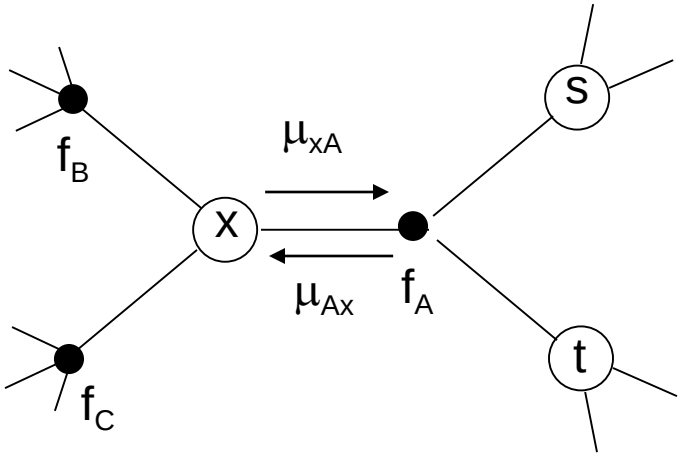
In the first algorithm (FB) the graph is converted into a tree (by choosing an arbitrary node as root) and the information is first sent from the leaves to the root and in the opposite way afterwards.

The second algorithm (PS) starts from a known configuration of the network. Every time new observations are available, the observed nodes sent information to their neighbors which propagate through the network until the are absorbed.

© Jorge Salvador Marques, 2002

Messages

There are two types of messages: μ_{xA}, μ_{Ax}



Product

$$\mu_{xA}(x) = \mu_{Bx}(x)\mu_{Cx}(x)$$

or

$$\mu_{xA}(x) = \delta(x, x_0)$$ if x is observed

Sum

$$\mu_{Ax}(x) = \sum_{s,t} f_A(x,s,t)\mu_{sA}(s)\mu_{tA}(t)$$

Note: each node propagates incoming messages in **all the other** directions. The beliefs are obtained by multiplying all the incomming messages of each node, nornalized by a multiplicative constant.

© Jorge Salvador Marques, 2002

Burgler Alarm

(Pearl)

Binary variables:

b – burglary

e - earthquake

a - alarm

Distributions:

P(b=1) = 0.1

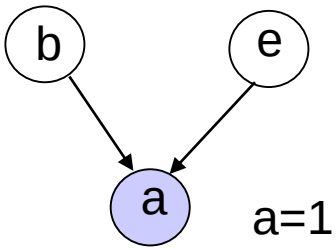
P(e=1) = 0.1

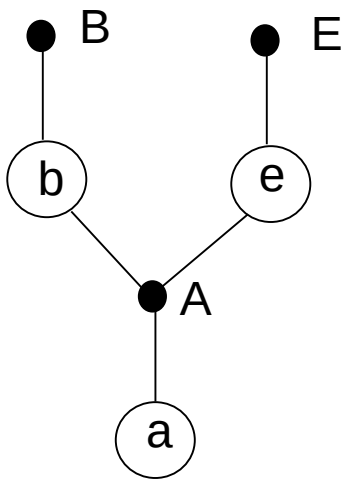
P(a=1 | b=0,e=0) = 0.001

P(a=1 | b=1,e=0) = 0.368

P(a=1 | b=0,e=1) = 0.135

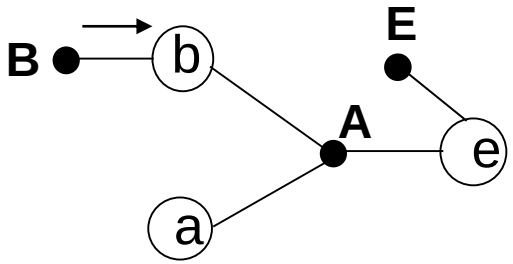
P(a=1 | b=1,e=1) = 0.607



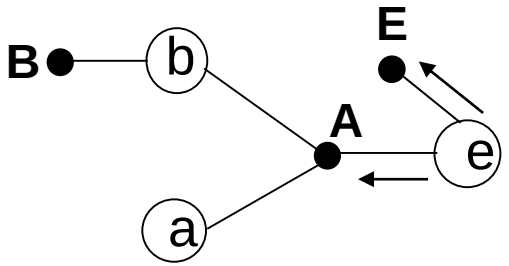


© Jorge Salvador Marques, 2002

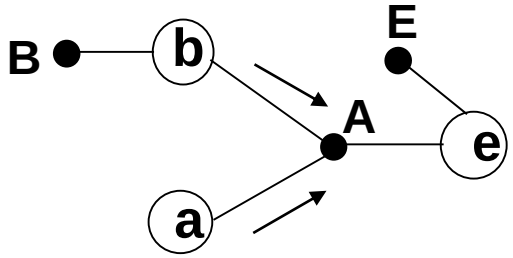
Burgler Alarm (2)



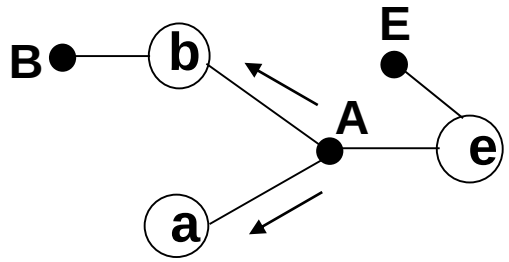
$\mu_{B\to b} = (0.9,0.1)$



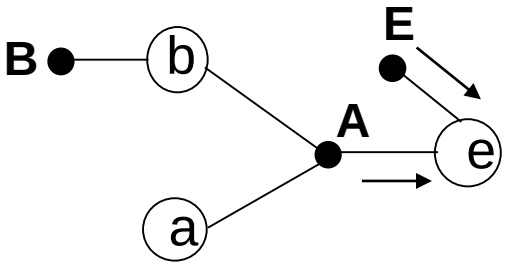
$\mu_{e\to A} = (0.9,0.1)$



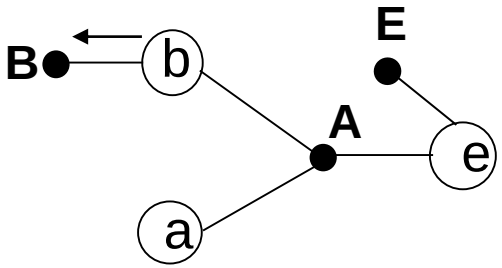
$\mu_{b\to A} = (0.9,0.1)$ $\mu_{a\to A} = (0,1)$



$\mu_{A\to b} = (0.0144,0.3919)$



$\mu_{A\to e} = (0.0377,0.1822)$ $\mu_{E\to e} = (0.9,0.1)$



$P(b/a=1) = (0.29,0.751)$ $P(\text{e/a}=1) = (0.651,0.349)$

© Jorge Salvador Marques, 2002

Monte Carlo

Monte Carlo method simulates the BN and computes statistics of the network variables from multiple realizations of x. The method is valid even in the presence of multiple paths.

The network simulation is based on the Gibbs sampler. The Gibbs sampler starts from an initial configuration and modifies a single variable x_i at a time. The new value of x_i is randomly selected according to the conditional distribution

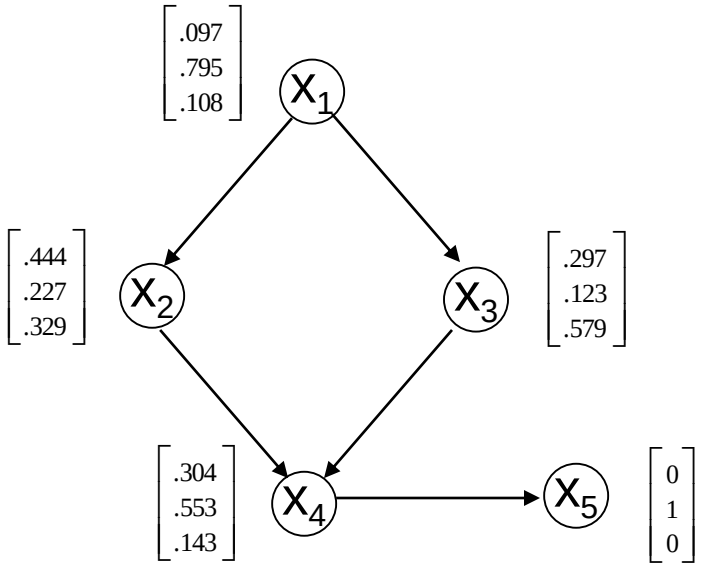
$$P(x_i / x \setminus \{x_i\}) = c \prod_k P(x_k / a_k)$$

This procedure is repeated until all the variables are updated. This task is repeated as many times as needed .

Note: to compute the conditional distribution, only the terms associated to the neighboring nodes are considered.

© Jorge Salvador Marques, 2002

Example



$p_{X_1}=[.1 \quad .8 \quad .1]'$

observations: $x_5=2$

$1\rightarrow2$	1	2	3	x_2
1	.2	.7	.1	
2	.4	.2	.4	
3	.9	.1	0	

x_1

$1\rightarrow3$	1	2	3	x_3
1	.8	.1	.1	
2	.2	.1	.7	
3	.6	.3	.1	

x_1

$23\rightarrow4$	1	2	3	x_4
11	.6	.3	.1	
12	.4	.1	.5	
13	.1	.8	.1	
21	.2	.6	.2	
22	.1	.5	.4	
23	.6	.3	.1	
31	.8	.1	.1	
32	.4	.3	.3	
33	.1	.8	.1	

x_2x_3

$4\rightarrow5$	1	2	3	x_5
1	.8	.1	.1	
2	.1	.1	.8	
3	.1	.1	.8	

x_4

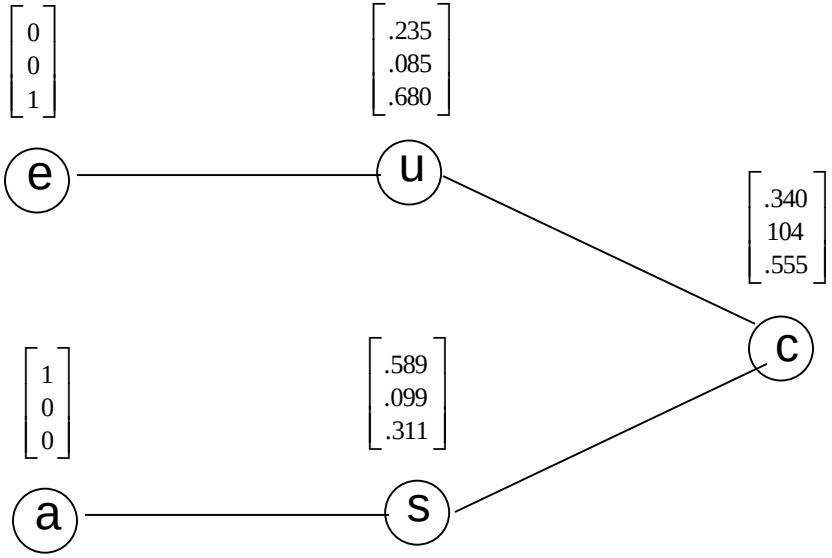
This example shows the beliefs obtained by Monte Carlo simulation using 10000 iterations.

(this graph has multiple paths)

© Jorge Salvador Marques, 2002

Sherlock Holmes (2)

The network was simulated 10000 times. All the variables were simulated starting from the previous configuration.



The mais suspect is the 3rd, although suspect 1 also has a high score.

© Jorge Salvador Marques, 2002

Marginalization

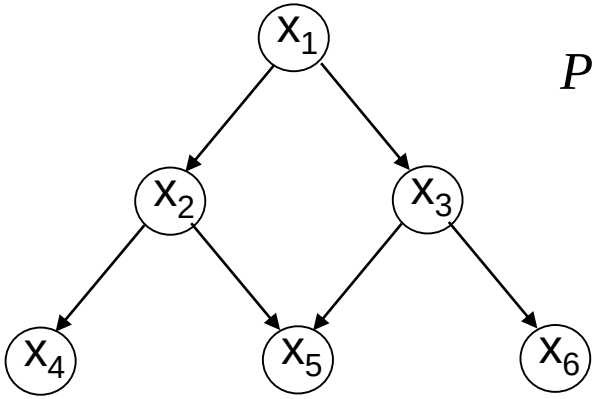
BN allow the use of efficient marginalization methods using the distributive property of multiplication. (*even in the presence of multiple paths*)

Several efficient algorithms are based on this property e.g., inference methods based on junction trees.

(the next slides are inspired in F. Jensen, Bayesian Networks and Decision Graphs, Springer, 2001)

© Jorge Salvador Marques, 2002

Example



$$P(x) = \underbrace{p(x_1)}_{\phi_1}\underbrace{p(x_2/x_1)}_{\phi_2}\underbrace{p(x_3/x_1)}_{\phi_3}\underbrace{p(x_4/x_2)}_{\phi_4}\underbrace{p(x_5/x_2,x_3)}_{\phi_5}\underbrace{p(x_6/x_3)}_{\phi_6}.$$

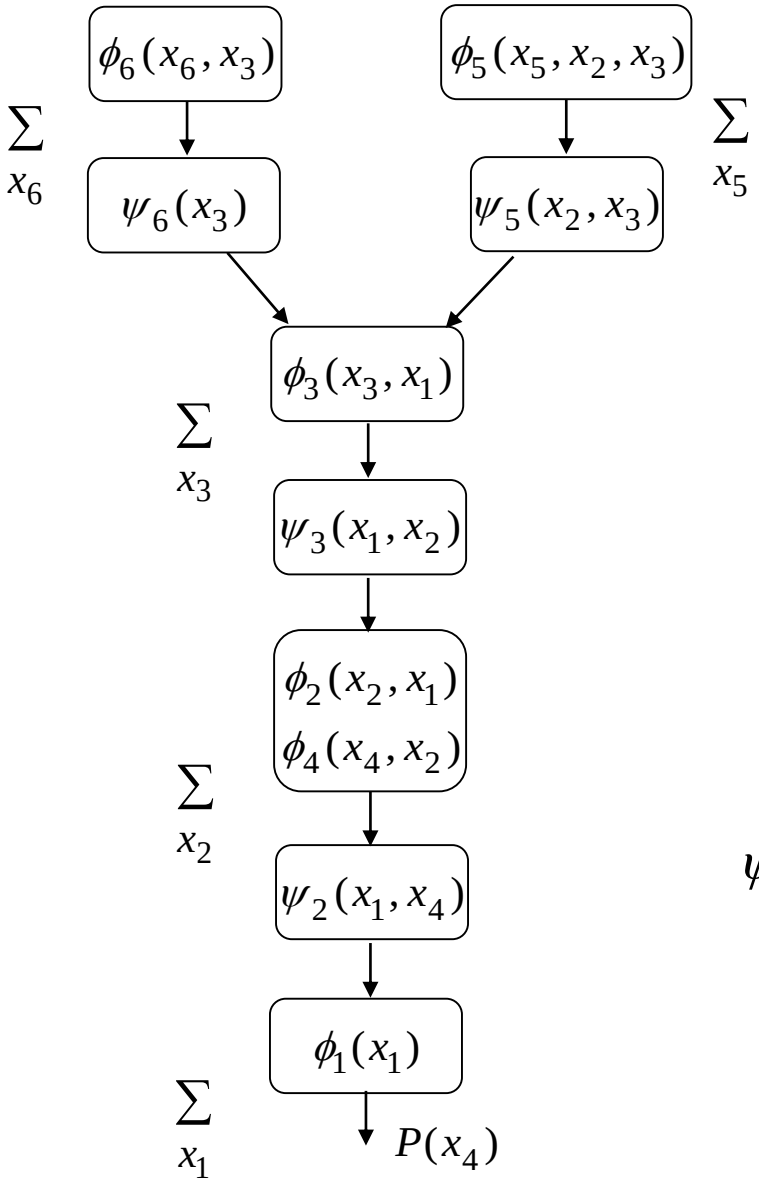
Problem: compute $P(x_4)$

$$\text{Direct method}: \quad P(x_4) = \sum_{x_1,x_2,x_3,x_5,x_6} P(x_1,x_2,x_3,x_4,x_5,x_6)$$

Efficient method:

$$P(x_4) = \sum_{x_1} p(x_1) \sum_{x_2} p(x_2/x_1)\, p(x_4/x_2) \sum_{x_3} p(x_3/x_1) \sum_{x_5} p(x_5/x_2,x_3) \sum_{x_6} p(x_6/x_3)$$

© Jorge Salvador Marques, 2002



$$\psi_6(x_3) = \sum_{x_6} \phi_6(x_6, x_3)$$

$$\psi_5(x_2, x_3) = \sum_{x_5} \phi_5(x_5, x_2, x_3)$$

$$\psi_3(x_1, x_2) = \sum_{x_3} \phi_3(x_3, x_1)\psi_6(x_3)\psi_5(x_2, x_3)$$

$$\psi_2(x_1, x_4) = \sum_{x_2} \psi_3(x_1, x_2)\phi_2(x_2, x_1)\phi_4(x_4, x_2)$$

$$P(x_4) = \sum_{x_1} \phi_1(x_1)\psi_2(x_1, x_4)$$

© Jorge Salvador Marques, 2002

Difficulty

The complexity of the marginalization operation depends on the choice of the elimination sequence.

Different elimination sequences may lead to different computational efforts.

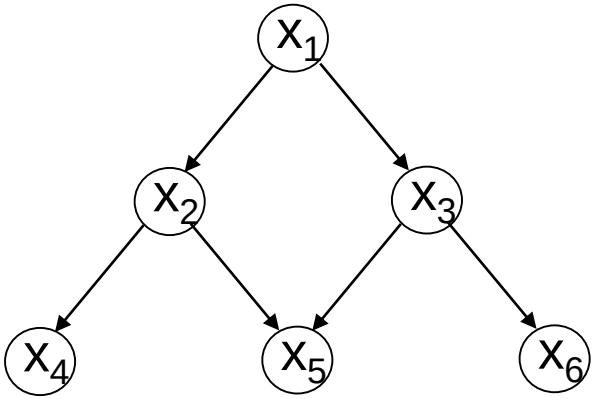
What is the best elimination sequence ?

© Jorge Salvador Marques, 2002

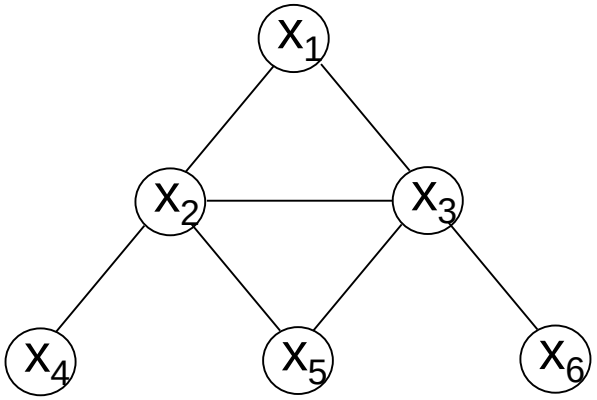
Domain Graph

The domain graph is an undirected graph with links among the variables of each potential function f_k.

Bayesian network



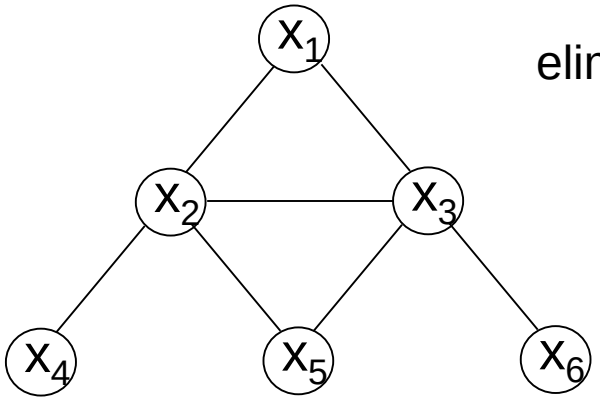
domain graph



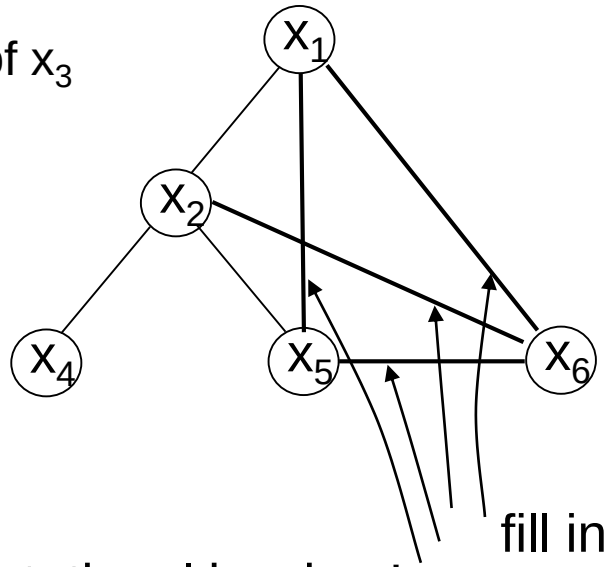
© Jorge Salvador Marques, 2002

Node Elimination

What happens to the domain graph when a variable is eliminated by marginalization ?



elimination of x_3



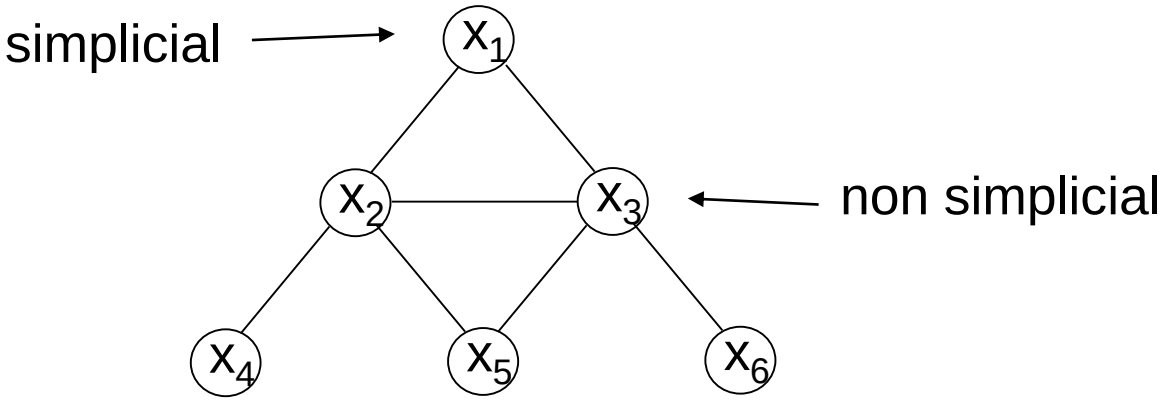
the fill ins (insertions) increase the computational burden !

© Jorge Salvador Marques, 2002

Simplicial Node

Definition: *simplicial node* is a node with a complete neighbor set.

Proposition: x is a simplicial node iif F_x is a clique.



No fill ins are created by the elimination of simplicial nodes !

© Jorge Salvador Marques, 2002

Triangulated Graphs

Is it possible to find an elimination sequence without fill ins ?

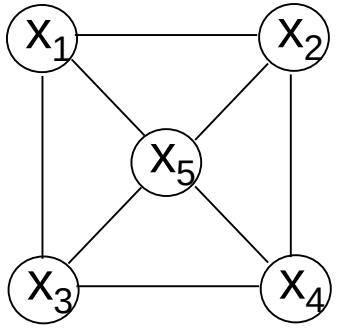
Theorem
If *G* is a triangulated graph, there is has a perfect elimination sequence (without fill ins) for each variable x_i.

Definition: an undirected graph G is a *triangulated graph* if there is one perfect elimination sequence i.e., without fill ins.

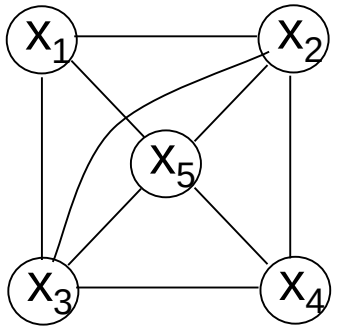
© Jorge Salvador Marques, 2002

Example

The concept of triangulated graph is misleading:



non triangulated graph



triangulated graph

© Jorge Salvador Marques, 2002

Triangulation of Graphs

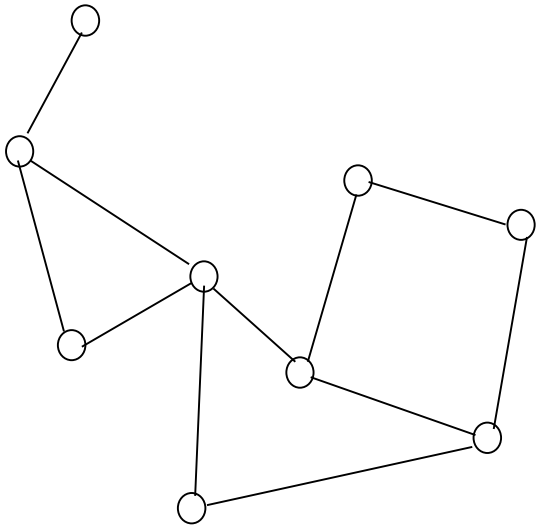
If G is a non triangulated graph, how can we compute a triangulated Graph with the minimum number of insertions ?

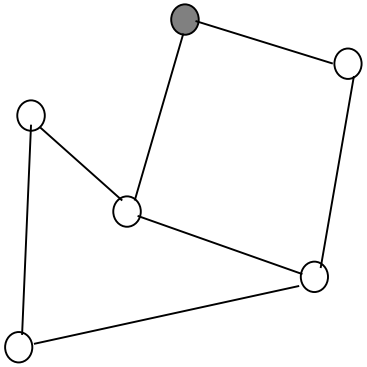
This is a NP-hard problem. There are sub-optimal solutions which usually work well.

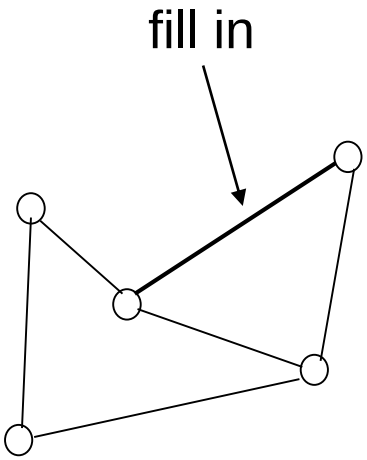
Algorithm:
Eliminate one node of G in each iteration, choosing a simplicial node. If there is not any simplicial node, chose a node x_i with the smallest neighborhood.

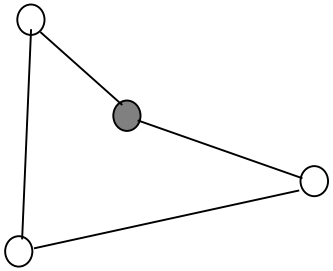
© Jorge Salvador Marques, 2002

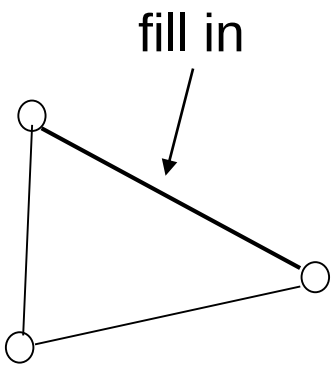
Example

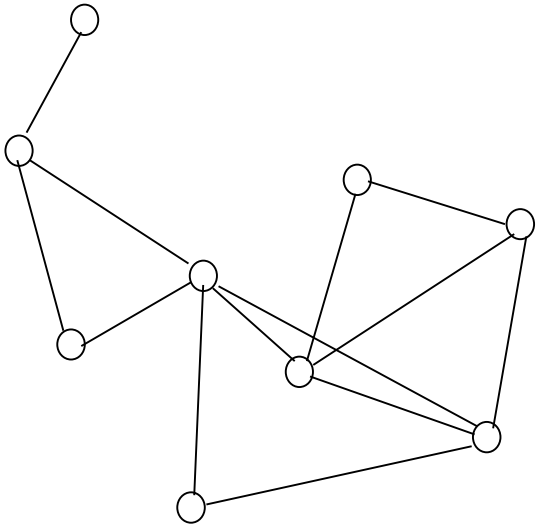












© Jorge Salvador Marques, 2002

Inference in Triangulated Graphs

How can we perform inference in a BN with a triangulated domain graph

Junction trees solve this problem for all the nodes, by using perfect elimination sequences i.e., without insertions.

Junction trees are trees in which the nodes are *cliques* of the domain graph.

© Jorge Salvador Marques, 2002

Computation of the Junction Tree

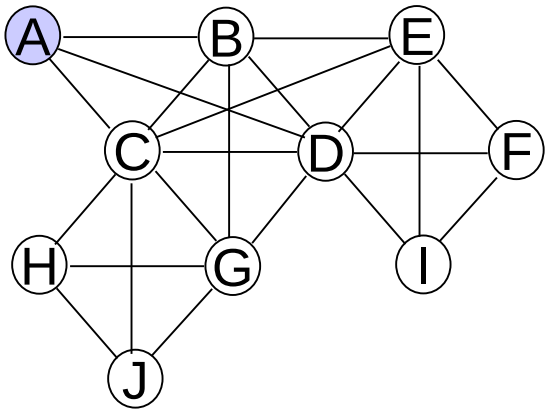
1. Compute cliques C_i and separator sets S_i (see below)
2. Link determination: sequentialy link each clique C_i to a clique C_i such that $S_i \subset C_j$ and $j>i$.
3. Determine the clique potentials: the potential of the ith clique j_i is the product of the potentials of the eliminated nodes.

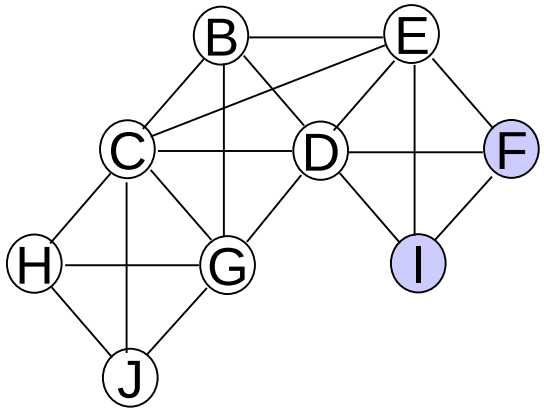
Clique Computation

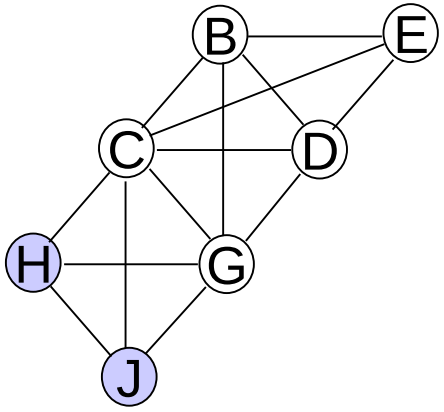
1. Choose a simplicial node x and define $C_i=F_x$.
2. Eliminate all the nodes of F_x which have all the neighbors in F_x; the other nodes define the separator S_i .
3. Repeat the previous steps and increase i, until all nodes are eliminated.

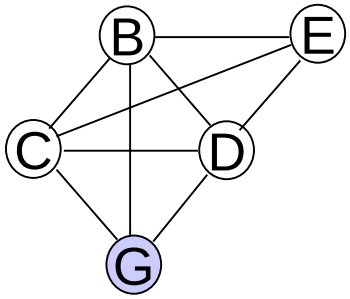
© Jorge Salvador Marques, 2002

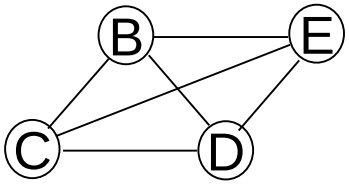
Example











C_1=ABCD S_1=BCD

C_2=DEFI S_2=DE

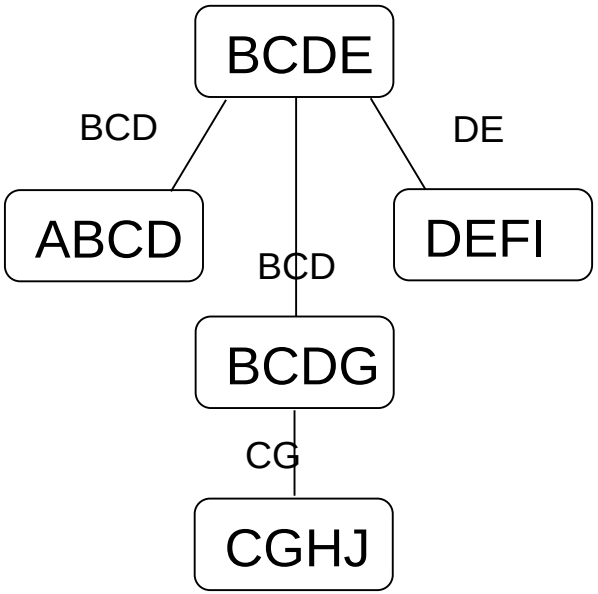
C_3=CGHJ S_3=CG

C_4=BCDG S_4=BCD

C_5=BCDE

© Jorge Salvador Marques, 2002

Example



C_1=ABCD S_1=BCD

C_2=DEFI S_2=DE

C_3=CGHJ S_3=CG

C_4=BCDG S_4=BCD

C_5=BCDE

$\varphi_1=\phi_A$

$\varphi_2=\phi_F\,\phi_I$

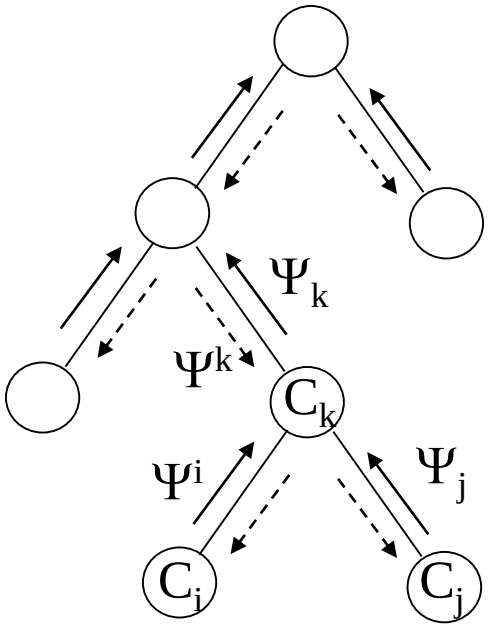
$\varphi_3=\phi_H\,\phi_J$

$\varphi_4=\phi_G$

$\varphi_5=\phi_B\,\phi_C\,\phi_D\,\phi_E$

© Jorge Salvador Marques, 2002

Inference in Junction Trees



Collect Evidence: propagate messages from the leaves to the root

$$\psi_k = (\varphi_k \psi_i \psi_j)^{\downarrow S_k}$$

Distribute Evidence: propagate messages from the root to the leaves

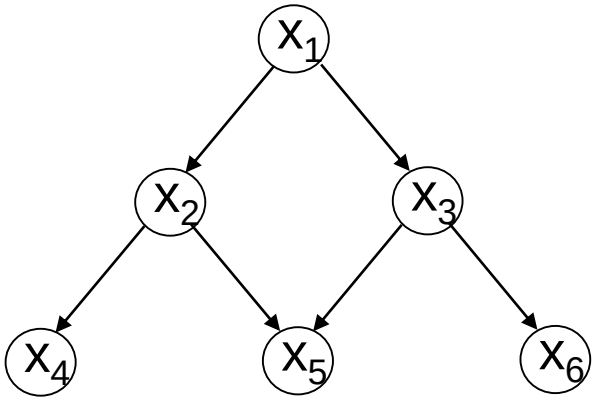
$$\psi^i = (\varphi_k \psi^k \psi_j)^{\downarrow S_i}$$

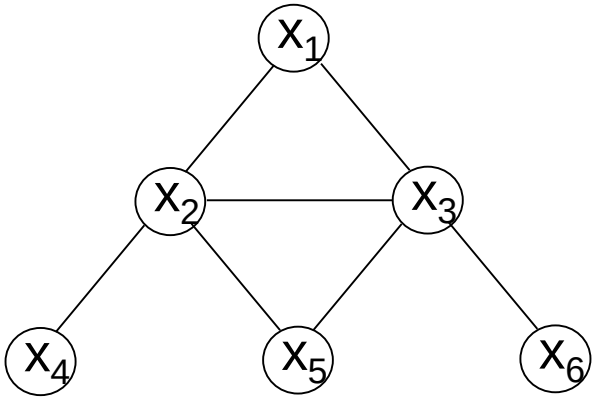
Notation: $()^{\downarrow x}$ marginalize with respect to x

$$P(C_k) = \varphi_k \psi^k \psi_i \psi_j$$

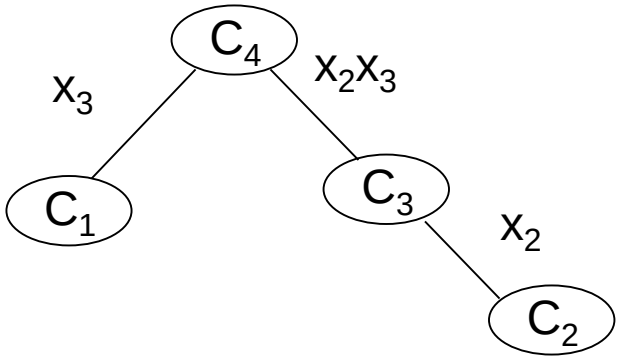
© Jorge Salvador Marques, 2002

Example





Junction tree



C_i	S_i	φ_i
$x_6\, x_3$	x_3	$\phi_6\,(x_6\,,x_3)$
$x_4\, x_2$	x_2	$\phi_4\,(x_4\,,x_2)$
$x_5\, x_2\, x_3$	$x_2\, x_3$	$\phi_5\,(x_5\,,x_2,x_3)$
$x_1\, x_2\ x_3$		$\phi_1\,(x_1)\ \phi_2\,(x_2\,,x_1)\ \phi_3\,(x_3\,,x_1)$

© Jorge Salvador Marques, 2002

Exemplo

Collecting evidence

$$\psi_2(x_2) = \sum_{x_4} \phi_4(x_4, x_2)$$

$$\psi_3(x_2, x_3) = \sum_{x_5} \phi_5(x_5, x_2, x_3)\psi_2(x_2)$$

$$\psi_1(x_3) = \sum_{x_6} \phi_6(x_6, x_3)$$

Distributing evidence

$$\psi^1(x_3) = \sum_{x_1, x_2} \phi_1(x_1)\phi_2(x_2, x_1)\phi_3(x_3, x_1)\psi_3(x_2, x_3)$$

$$\psi^3(x_2, x_3) = \sum_{x_1} \phi_1(x_1)\phi_2(x_2, x_1)\phi_3(x_3, x_1)\psi_1(x_3)$$

$$\psi^2(x_2) = \sum_{x_2, x_5} \phi_5(x_5, x_2, x_3)\psi^3(x_2, x_3)$$

$$P(C_1) = P(x_3, x_6) = \phi_6(x_6, x_3)\psi^1(x_3)$$

$$P(C_3) = P(x_2, x_3, x_5) = \phi_5(x_5, x_2, x_3)\psi^3(x_2, x_3)\psi_2(x_4)$$

© Jorge Salvador Marques, 2002

Software

Software for Bayesian Networks:

- Bayes Net Toolbox (Murphy)
- BAIES (Cowell, 1995)
- Hugin (Andersen, 1989)
- IDEAL (Srinvas, Breese, 1990)
- PRESS (Gammerman, 1995)

© Jorge Salvador Marques, 2002

Comment

"Belief networks are designed and trained to answer more than just the question of classifying future cases. They are able to give a much higher level of explanation, including exploring what were important input features in reaching the conclusion and whether the input data were in some sense in conflict. To do so they model the whole joint distribution. Although there will be an advantage in using qualitative knowledge (at least if it is a reasonable approximation to reality), the need to model the whole distribution makes more demands on limited data resources"
Ripley, 1996

© Jorge Salvador Marques, 2002

Exercises

1. Describe a problem and define a Bayesian network to solve it.
2. Solve the Sherlock Holmes problem using Frey algorithm and junction trees.

© Jorge Salvador Marques, 2002

Project

Design a system for medical diagnosis of two diseases A, B from two medical exams C, D or any subset of these.

Training data:

A	0	1	1	0	0	1	0	0	1	0	1	0	0	0	1	1	0	0	1
B	0	0	1	1	0	0	1	1	1	0	0	1	1	0	1	0	1	1	1
C	0	0	1	1	0	0	1	1	1	0	0	1	1	0	1	0	1	1	1
D	1	1	1	0	0	1	1	0	1	0	1	0	1	0	1	1	0	1	0

Characterize the output of the system for all input configurations.
Characterize the performance of the system on the training set.

Note: the assessment of the system performance should be based on independent data.

© Jorge Salvador Marques, 2002

Bibliografia

• Z. Ghahramani, An Introduction to Hidden Markov Models and Bayesian Networks, 2001.

•F. Jensen, Bayesian Networks and Decision Graphs, New York/Berlin, Springer, 2001.

•J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 2ª ed. Revista, 1997.

•B. Frey, Graphical Models for Machine Learning and Digital Communication, MIT Press 1998.

•B. D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, 1996.

© Jorge Salvador Marques, 2002