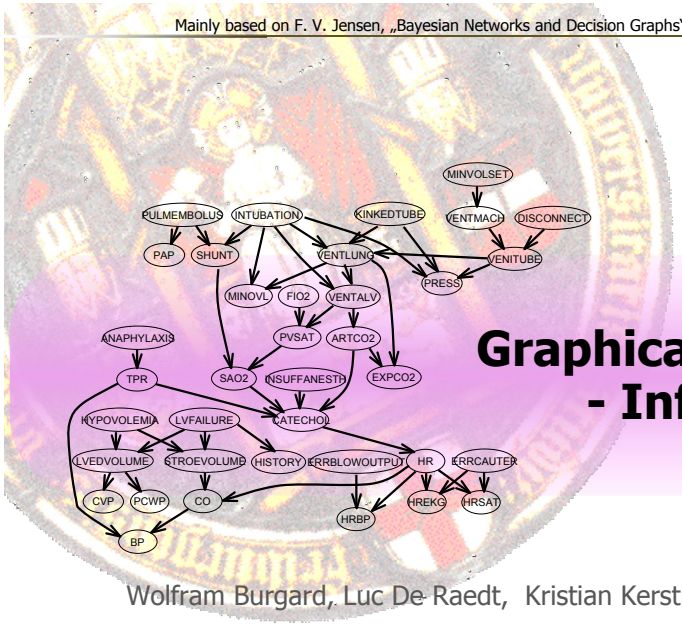
Mainly based on F. V. Jensen, „Bayesian Networks and Decision Graphs", Springer-Verlag New York, 2001.

Advanced I WS 06/07



Graphical Models - Inference -

Junction Tree

Wolfram Burgard, Luc De Raedt, Kristian Kersting, Bernhard Nebel

Albert-Ludwigs University Freiburg, Germany

Outline

- Introduction
- Reminder: Probability theory
- Basics of Bayesian Networks
- Modeling Bayesian networks
- Inference (VE, Junction tree)
- Excourse: Markov Networks
- Learning Bayesian networks
- Relational Models

Why Junction trees ?

- Multiple inference, e.g.,

$$P(X_i, e) \qquad P(X_i|e) = \frac{P(X_i, e)}{P(e)}$$

- For each i do variable elimination?

No, instead reuse information resp. computations

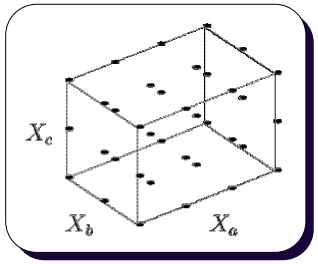
Potentials

A **potential** ϕ_A over a set of variables A is a function that maps each configuration into a **non-negative real number**.

$\text{dom}\phi_A = A$ (domain of ϕ_A)

Examples: Conditional probability distribution and joint probability distributions are special cases of potentials

Potentials



Ex: A potential $\phi_{A,B,C}$ over the set of variables {A,B,C}. A has four states, and B and C has three states.
dom$\phi_{A,B,C}$ ={A,B,C}

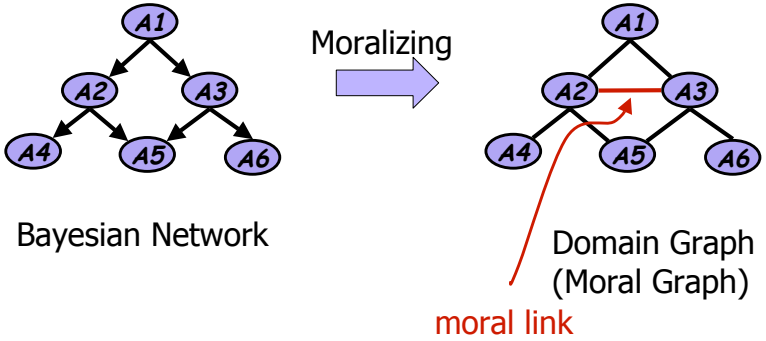
VE using Potentials

Let Φ be a set of potentials (e.g. CPDs), and let X be a variable. X is eliminated from Φ:

1. Remove all potentials in Φ with X in the domain. Let Φ_X that set.
2. Calculate $\Phi^{-X} = \sum_X \prod \Phi_X$
3. Add Φ^{-X} to Φ
4. Iterate

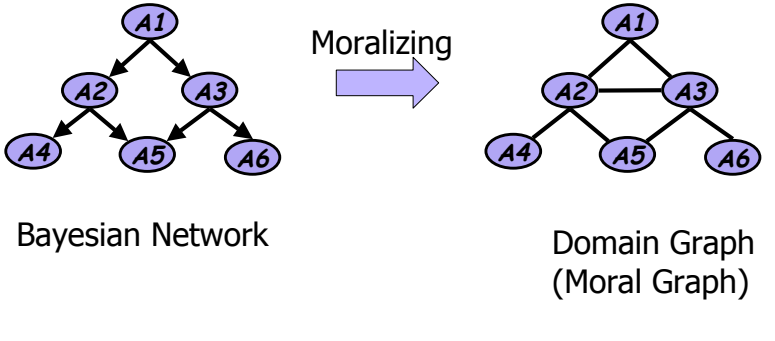
Potential where X is not a member of the domain

Domain Graph

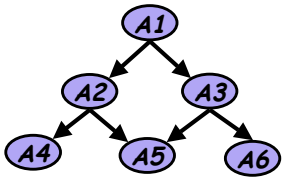


Let $\Phi=\{\phi_1,...,\phi_n\}$ be potentials over $U=\{A_1,...,A_m\}$ with $dom\phi_i=D_i$. The domain graph for Φ is the undirected graph with variables of U as nodes and with a link between pairs of variables being members of the same D_i.

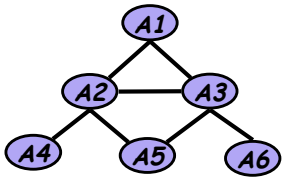
Moralizing



Moralizing



Moralizing



Bayesian Network

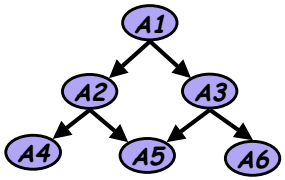
Domain Graph
(Moral Graph)

CPDs

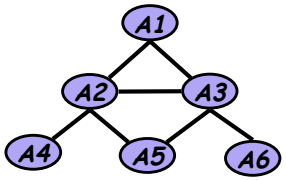
$$P(A_1)$$
$$P(A_2|A_1)$$
$$P(A_3|A_1)$$
$$P(A_4|A_2)$$
$$P(A_5|A_2, A_3)$$
$$P(A_6|A_3)$$

Advanced I WS 06/07

Moralizing



Moralizing



Bayesian Network

Domain Graph
(Moral Graph)

CPDs

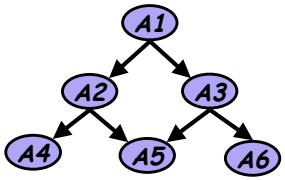
$$P(A_1)$$
$$P(A_2|A_1)$$
$$P(A_3|A_1)$$
$$P(A_4|A_2)$$
$$P(A_5|A_2, A_3)$$
$$P(A_6|A_3)$$

CPD
=
potential

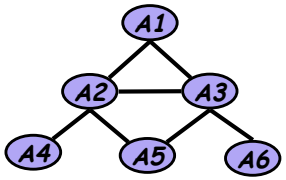
$$dom(\phi_1) = \{A_1\}$$
$$dom(\phi_2) = \{A_2, A_1\}$$
$$dom(\phi_3) = \{A_3, A_1\}$$
$$dom(\phi_4) = \{A_4, A_2\}$$
$$dom(\phi_5) = \{A_5, A_2, A_3\}$$
$$dom(\phi_6) = \{A_6, A_3\}$$

Advanced I WS 06/07

Moralizing



Moralizing



Bayesian Network

Potentials induce
undirected edges

Domain Graph
(Moral Graph)

CPDs

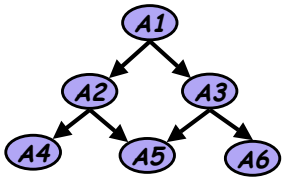
$$P(A_1)$$
$$P(A_2|A_1)$$
$$P(A_3|A_1)$$
$$P(A_4|A_2)$$
$$P(A_5|A_2, A_3)$$
$$P(A_6|A_3)$$

CPD
=
potential

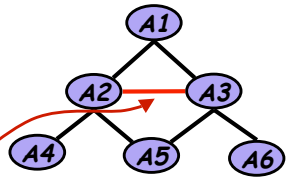
$$dom(\phi_1) = \{A_1\}$$
$$dom(\phi_2) = \{A_2, A_1\}$$
$$dom(\phi_3) = \{A_3, A_1\}$$
$$dom(\phi_4) = \{A_4, A_2\}$$
$$dom(\phi_5) = \{A_5, A_2, A_3\}$$
$$dom(\phi_6) = \{A_6, A_3\}$$

Advanced I WS 06/07

Moralizing



Moralizing



moral link

Bayesian Network

Potentials induce
undirected edges

Domain Graph
(Moral Graph)

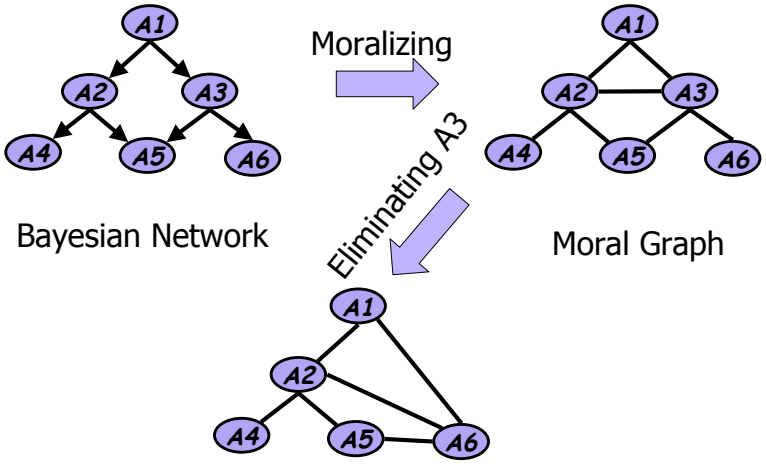
CPDs

$$P(A_1)$$
$$P(A_2|A_1)$$
$$P(A_3|A_1)$$
$$P(A_4|A_2)$$
$$P(A_5|A_2, A_3)$$
$$P(A_6|A_3)$$

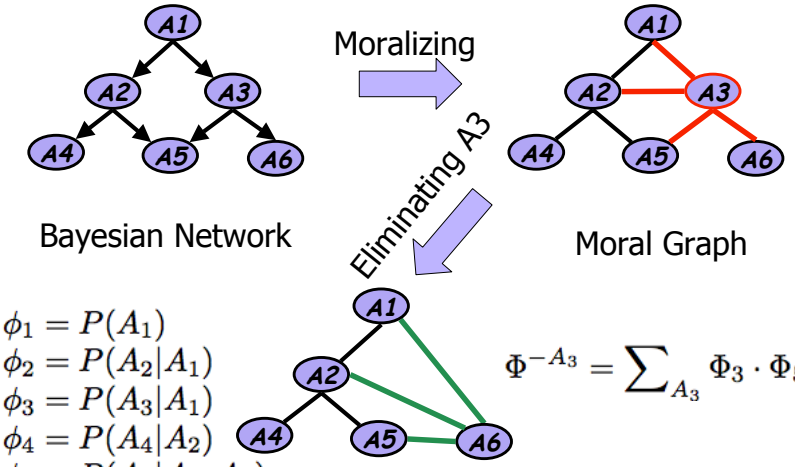
CPD
=
potential

$$dom(\phi_1) = \{A_1\}$$
$$dom(\phi_2) = \{A_2, A_1\}$$
$$dom(\phi_3) = \{A_3, A_1\}$$
$$dom(\phi_4) = \{A_4, A_2\}$$
$$dom(\phi_5) = \{A_5, A_2, A_3\}$$
$$dom(\phi_6) = \{A_6, A_3\}$$

Elimination Sequence



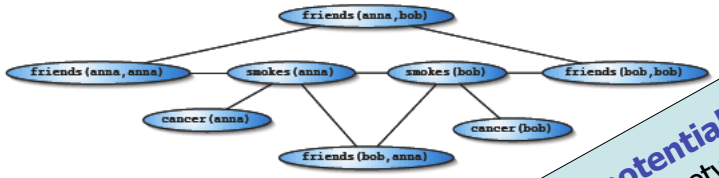
Elimination Sequence



$$\phi_1 = P(A_1)$$
$$\phi_2 = P(A_2|A_1)$$
$$\phi_3 = P(A_3|A_1)$$
$$\phi_4 = P(A_4|A_2)$$
$$\phi_5 = P(A_5|A_2, A_3)$$
$$\phi_6 = P(A_6|A_3)$$

$$\Phi^{-A_3} = \sum_{A_3} \Phi_3 \cdot \Phi_5 \cdot \Phi_6$$

Reminder: Markov Networks

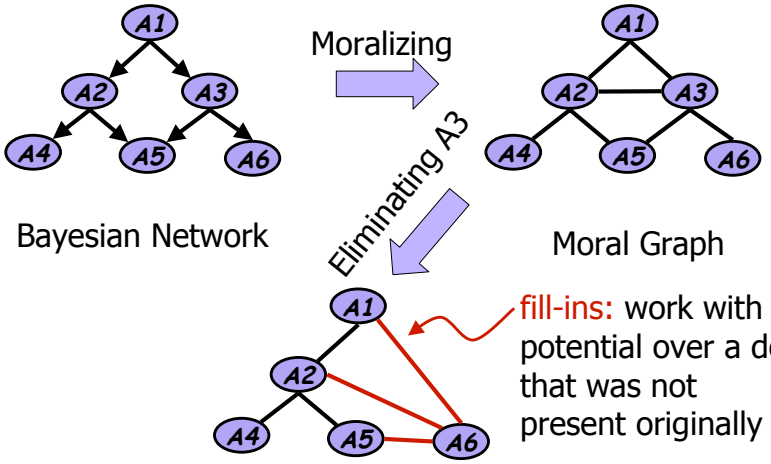


VE on potentials works for Markov networks

- Undirected Graphs $X_1, \ldots, X_n$
- Nodes = random variables
- Cliques = potentials (~ local jpd) ϕ_k

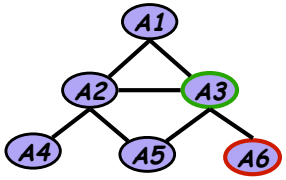
$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$$

Elimination Sequence



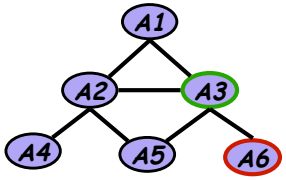
GOAL:Elimination sequence that does not introduce fill-ins

Perfect Elimination Sequence ...

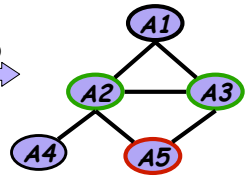


A6

Perfect Elimination Sequence ...

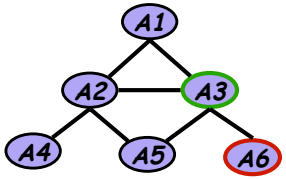


A6

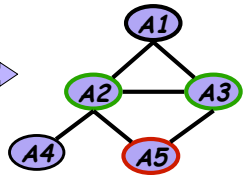


A5

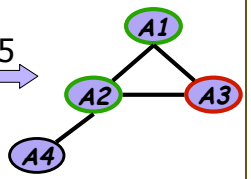
Perfect Elimination Sequence ...



A6

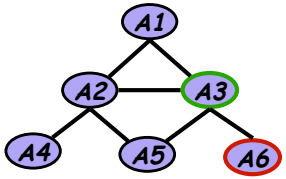


A5

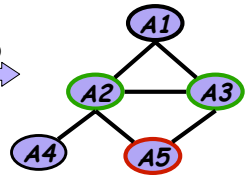


A3

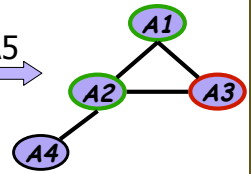
Perfect Elimination Sequence ...



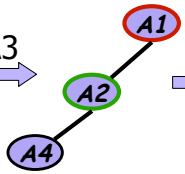
A6



A5

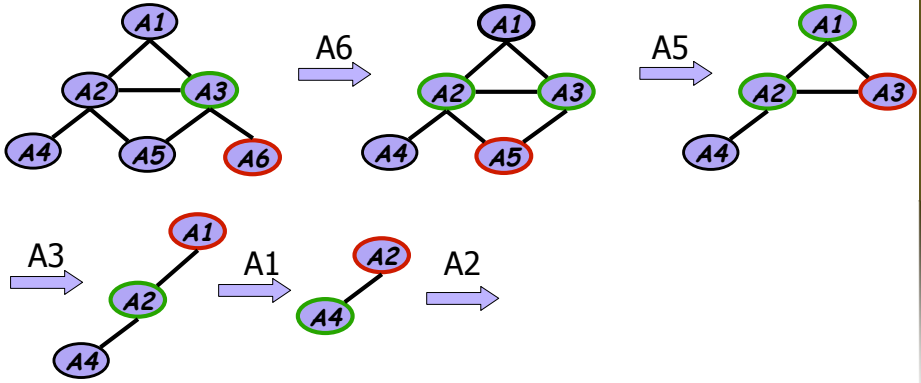


A3

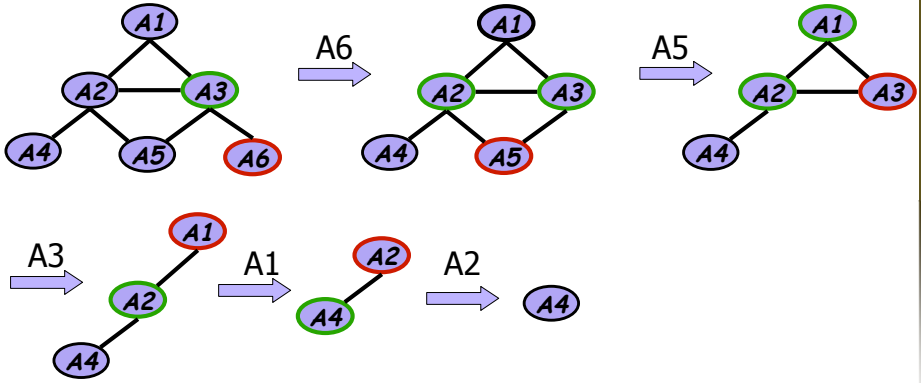


A1

Perfect Elimination Sequence ...

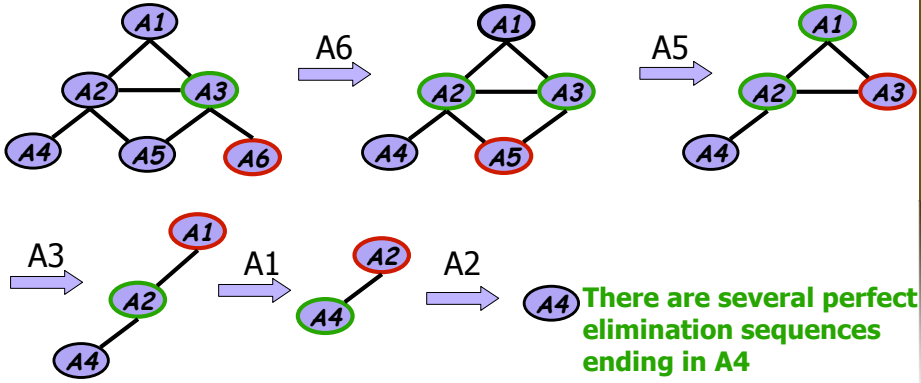


Perfect Elimination Sequence ...



Perfect Elimination Sequence ...

... do not introduce fill-ins

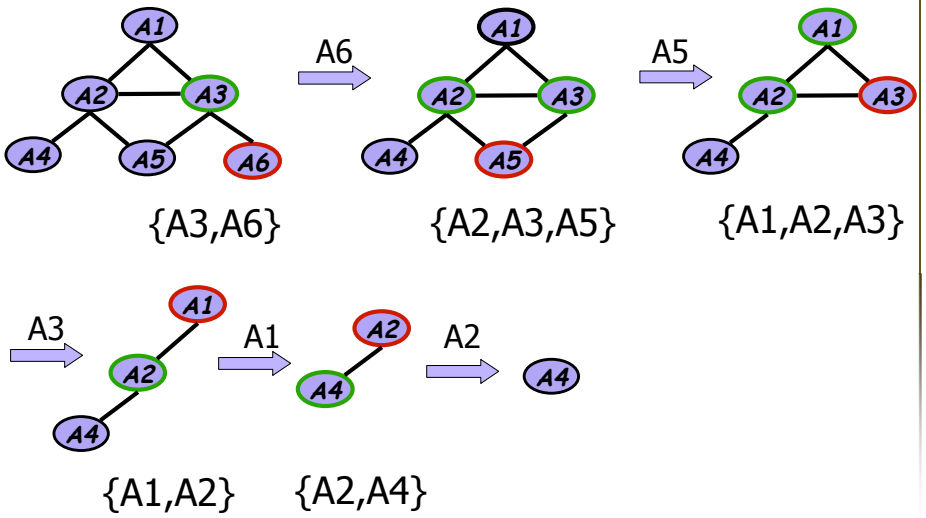


There are several perfect elimination sequences ending in A4
But they have different complexities

Complexity of Elimination Sequence

- Characterized by the set of domains
- Set of domains of potentials produced during the elimination (potentials that are subsets of other potentials are removed).
- A6,A5,A3,A1,A2,A4:
  {{A6,A3},{A2,A3,A5},{A1,A2,A3},{A1,A2},{A2,A4}}

Complexity of Elimination Sequence



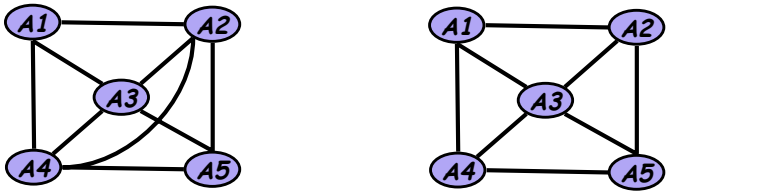
Complexity of Elimination Sequence

- All perfect elimination sequences produce the same domain set, namely the set of cliques of the domain

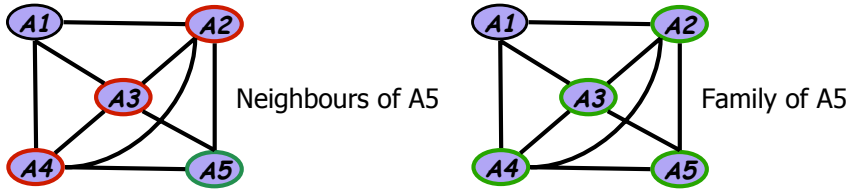
- ⟹ Any **perfect elimination sequence** ending with A is **optimal** with respect to **computing P(A)**

Triangulated Graphs and Join Trees

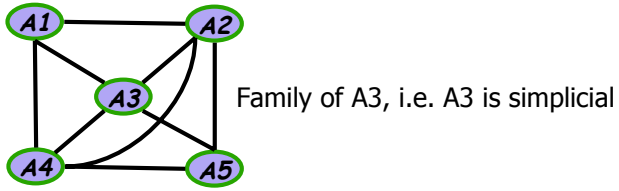
- An undirected graph with **complete elimination sequence** is called a *triangulated* graph



Triangulated Graphs and Join Trees



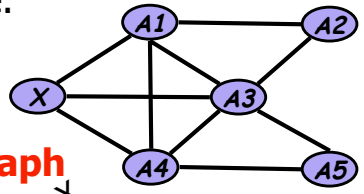
- Complete neighbour set = simplicial node
- X is simplicial iff family of X is a clique

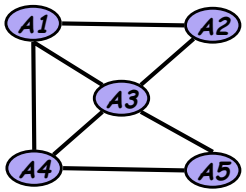


Triangulated Graphs and Join Trees

Let **G** be a **triangulated** graph, and let **X** be a **simplicial** node. Let G´be the graph resulting from **eliminating X** from G.

Then G´is a triangulated graph





Eliminating X

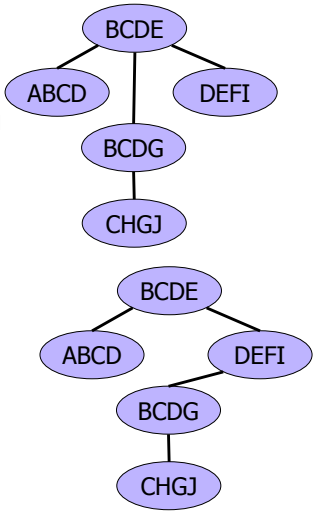
Triangulated Graphs and Join Trees

- A triangulated graph with at least two nodes has at least two simplicial nodes
- In a triangulated graph, each variable A has a perfect elimination sequence ending with A
- Not all domain graphs are triangulated: An undirected graph is triangulated iff all nodes can be eliminated by successively eliminating a simplicial node

Join Trees

Let G be the set of cliques from an undirected graph, and let the cliques of G be organized in a tree.

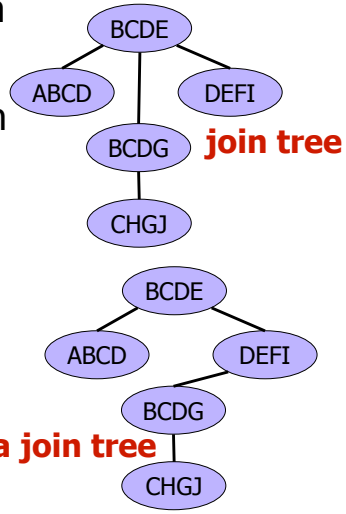
- T is a **join tree** if **for any pair of nodes V, W all nodes on the path between V an W contain the intersection V∩W**



Join Trees

Let G be the set of cliques from an undirected graph, and let the cliques of G be organized in a tree.

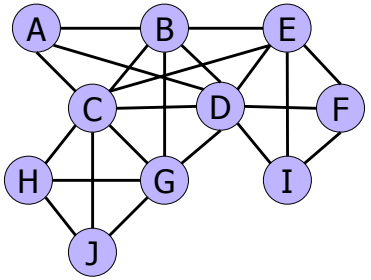
- T is a **join tree** if **for any pair of nodes V, W all nodes on the path between V an W contain the intersection V∩W**



Join Trees

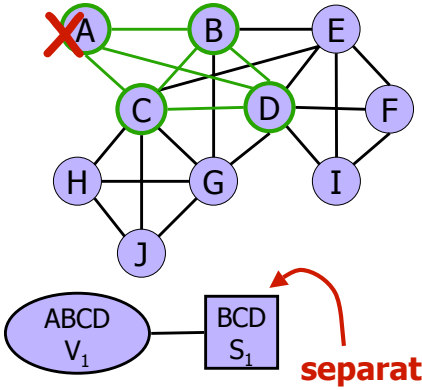
- If the cliques of an undirected graph G can be organized into a join tree, then G is triangulated
- If the undirected graph is triangulated, then the cliques of G can be organizes into a join tree

Triangulated, undirected Graph -> Join trees



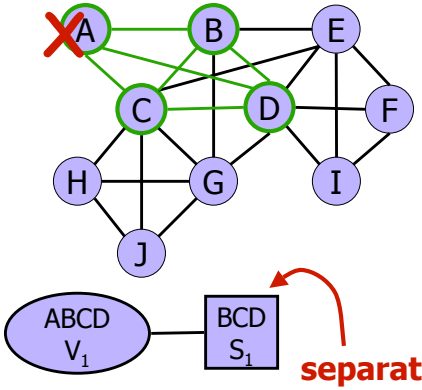
- Simplicial node X
- Family of X is a clique
- Eliminate nodes from family of X which have only neighbours in the family of X
- Give family of X is number according to the number of nodes eliminated
- Denote the set of remaining nodes S_i

Triangulated, undirected Graph -> Join trees



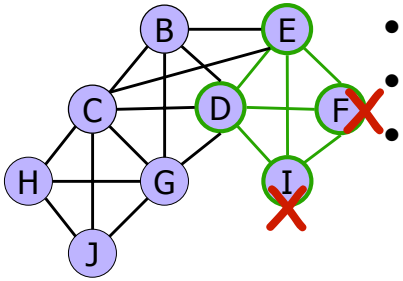
- Simplicial node X
- Family of X is a clique
- Eliminate nodes from family of X which have only neighbours in the family of X
- Give family of X a number i according to the number of nodes eliminated
- Denote the set of remaining nodes S_i

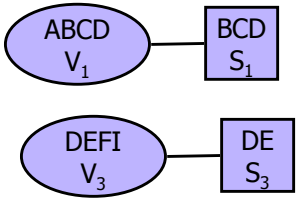
Triangulated, undirected Graph -> Join trees



- Simplicial node X **A** **{A,B,C,D}**
- Family of X is a clique
- Eliminate nodes from family of X which have only neighbours in the family of X **{B,C,D}**
- Give family of X a number i according to the number of nodes eliminated **1**
- Denote the set of remaining nodes S_i **S_1**

Triangulated, undirected Graph -> Join trees

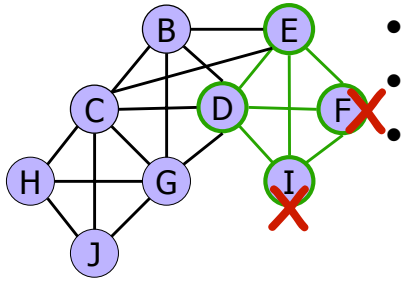


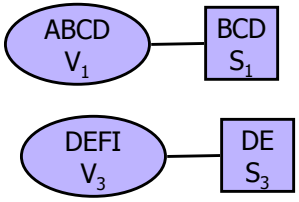


- Simplicial node X
- Family of X is a clique
- Eliminate nodes from family of X which have only neighbours in the family of X
- Give family of X is number according to the number of nodes eliminated
- Denote the set of remaining nodes S_i

Advanced I WS 06/07

Triangulated, undirected Graph -> Join trees



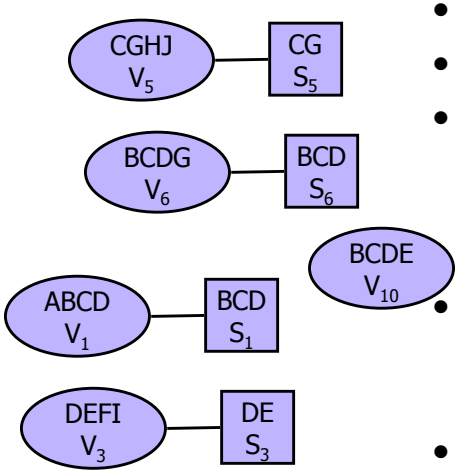


- Simplicial node X **E** **{D,E,F,I}**
- Family of X is a clique
- Eliminate nodes from family of X which have only neighbours in the family of X **{D,E}**
- Give family of X is number according to the number of nodes eliminated **3**
- Denote the set of remaining nodes S_i **S_3**

Bayesian Networks - Inference (Junction Tree)

Advanced I WS 06/07

Triangulated, undirected Graph -> Join trees

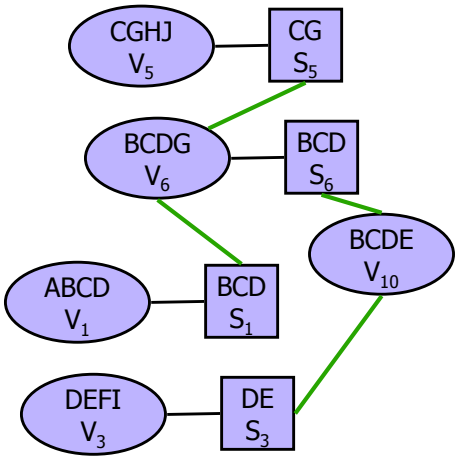


- Simplicial node X
- Family of X is a clique
- Eliminate nodes from family of X which have only neighbours in the family of X
- Give family of X is number according to the number of nodes eliminated
- Denote the set of remaining nodes S_i

Bayesian Networks - Inference (Junction Tree)

Advanced I WS 06/07

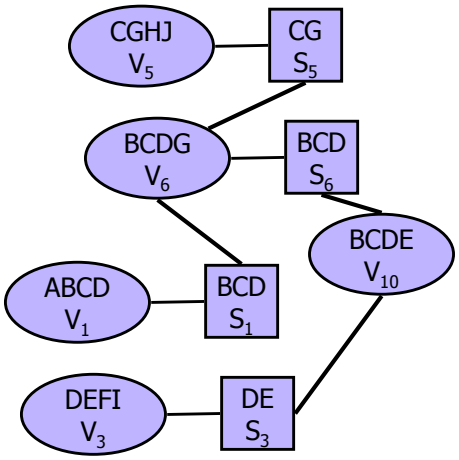
Triangulated, undirected Graph -> Join trees



- Connect each separator S_i to a clique V_j, j>i, such that S_i is a subset of V_j

Bayesian Networks - Inference (Junction Tree)

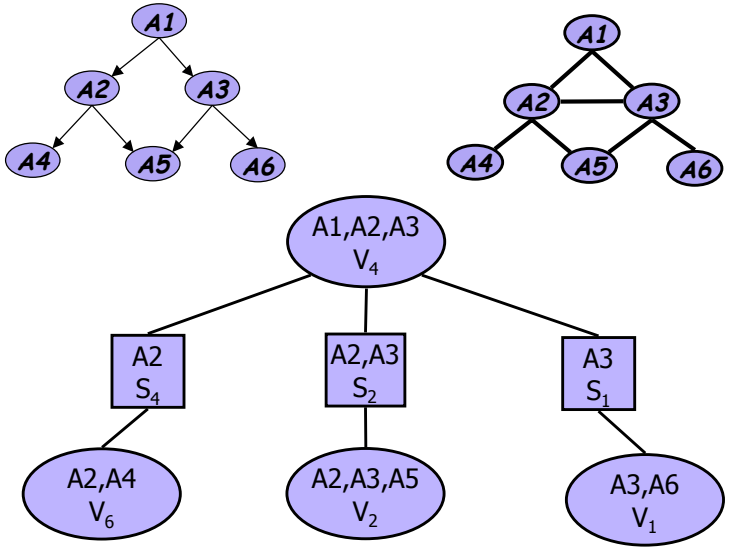
Join Tree



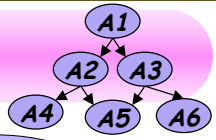
The potential representation of a join tree (aka clique tree) is the product of the clique potentials, dived by the product of the separator potentials.

$$P(\mathbf{X}) = \frac{\prod_c \phi_c(\mathbf{X})}{\prod_s \phi_s(\mathbf{X})}$$

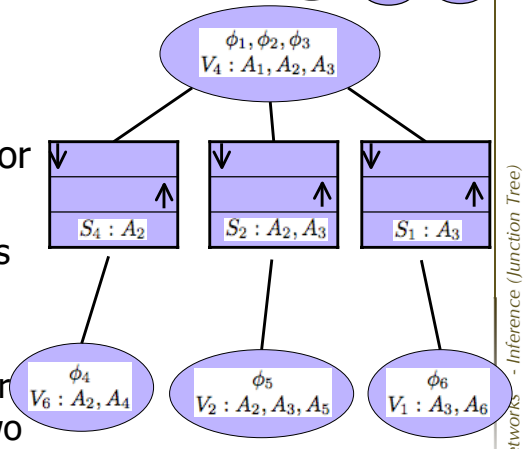
Yet Another Example



Junction Trees



- Let Φ be a set of potentials with a triangulated domain graph. A junction tree for Φ is join tree for G with
  - Each potential ϕ in Φ is associated to a clique containing dom(ϕ)
  - Each link has separator attached containing two mailboxes, one for each direction

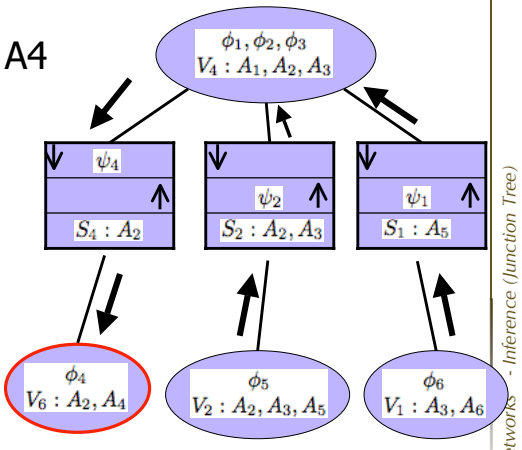


Propagation on a Junction Tree

- Node V can send exactly one message to a neighbour W, and it may only be sent when V has received a message from all of its other neighbours
- Choose one clique (arbitrarily) as a root of the tree; collect message to this node and then distribute messages away from it.
- After collection and distribution phases, we have all we need in each clique to compute potential for variables.

Junction Trees - Collect Evidence

- P(A4) ?
- Find clique containing A4
- V6 temporary root
- Send messages from leaves to root
- V4 assembles the incoming messages, potential



Junction Tree - Messages

- Propagation/message passing between two adjacent cliques C_1, C_2 (S_0 is their seperator)
  - Marginalize C_1's potential to get new potential for S_0

$$\phi^*_{S_o} = \sum_{C_1 \setminus S_0} \phi_{C_1}$$

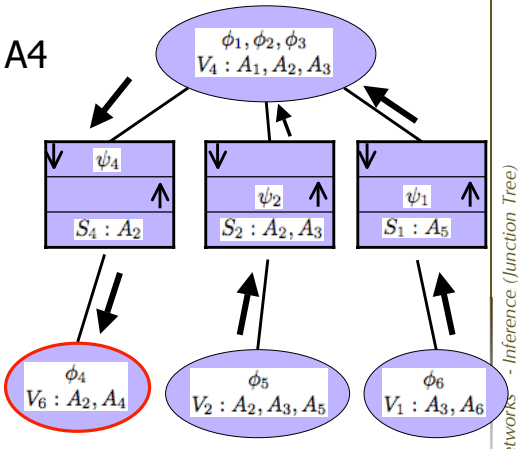
  - Update C_2's potential

$$\phi^*_{C_2} = \phi_{C_2} \frac{\phi^*_{S_0}}{\phi_{S_0}}$$

  - Update S_0's potential to its new potential

Junction Trees - Collect Evidence

- P(A4) ?
- Find clique containing A4
- V6 temporary root
- Send messages from leaves to root
- V4 assembles the incoming messages, potential



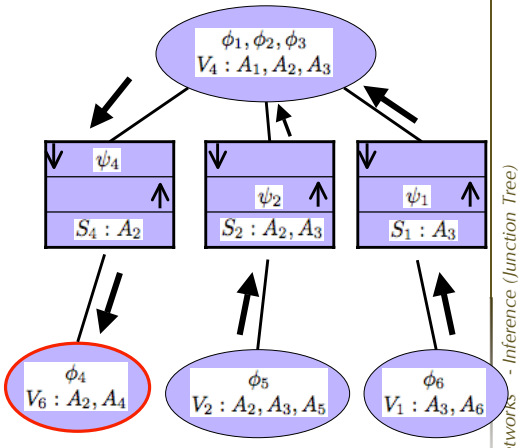
$$\psi_4 = \sum_{A_1} \phi_1 \cdot \phi_2 \sum_{A_3} \phi_3 \cdot \psi_2 \cdot \psi_1$$

Junction Trees - Collect Evidence

- P(A4) ?

$$P(A_4) = \sum_{A_2} \psi_4 \cdot \phi_4$$

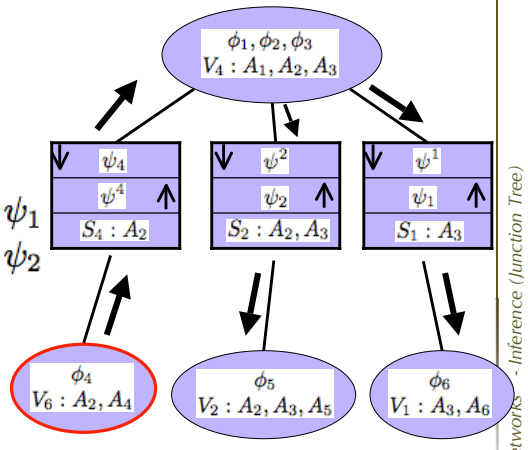
- All marginals?



Junction Trees - Distribute Evidence

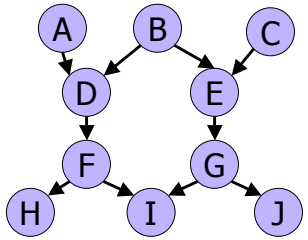
- All marginals?

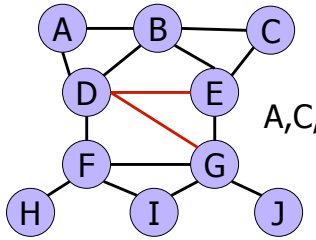
$$\psi^4 = \sum_{A_2} \phi_4$$
$$\psi^2 = \psi^4 \cdot (\sum_{A_1} \phi_1\phi_2\phi_3) \cdot \psi_1$$
$$\psi^1 = \psi^4 \cdot (\sum_{A_1} \phi_1\phi_2\phi_3) \cdot \psi_2$$
$$P(A_3) = \sum_{A_6} \phi_6 \cdot \psi^1$$
$$P(A_6) = \sum_{A_3} \phi_6 \cdot \psi^1$$
$$\ldots$$



Nontriangulated Domain Graphs

- Embed domain graph in a traingulated graph
- Use ist junction tree
- Simple idea:
  - Eliminate variables in some order
  - If you wish to eliminate a node with non-complete neighbour set, make it complete by adding fill-ins





A,C,H,I,J,B,G,D,E,F

Summary JTA

- Convert Bayesian network into JT
- Initialize potentials and separators
- Incorporate Evidence (set potentials accordingly)
- Collect and distribute evidence
- Obtain clique marginals by marginalization/normalization

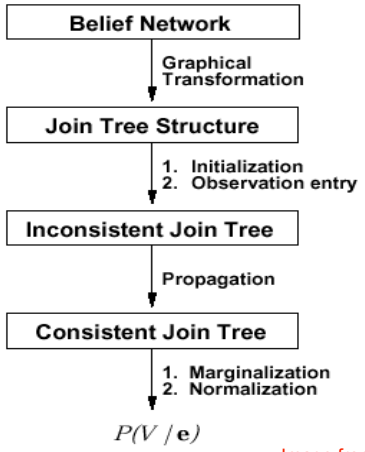


Image from Cecil Huang

Inference Engines

- (Commercial) HUGIN : http://www.hugin.com
- (Commercial) NETICA: http://www.norsys.com
- Bayesian Network Toolbox for Matlab www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html
- GENIE/SMILE (JAVA) http://www2.sis.pitt.edu/~genie/
- MSBNx, Microsoft, http://research.microsoft.com/adapt/MSBNx/
- Profile Toolbox http://www.informatik.uni-freiburg.de/~kersting/profile/