

**New reading:  
Chapter 7 of Koller&Friedman**

# Clique trees 2

Graphical Models – 10708

Carlos Guestrin

Carnegie Mellon University

October 3<sup>rd</sup>, 2005

# Announcements



- **Homework 2:**

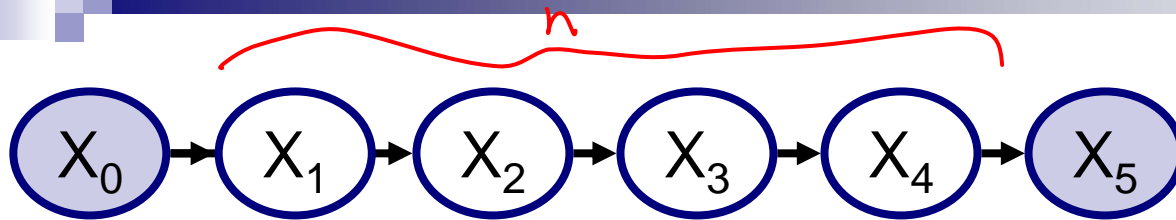
- Out today/tomorrow

- Programming part in groups of 2-3

- **Class project**

- More details on Wednesday

What if I want to compute  
 $P(X_i | x_0, x_{n+1})$  for each  $i$ ?

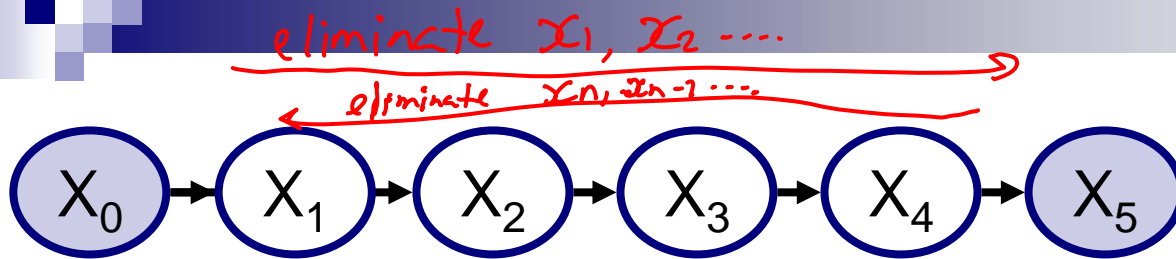


**Compute:**  
 $P(X_i | x_0, x_{n+1})$

Variable elimination for each  $i$ ?  $O(n)$

Variable elimination for ~~each~~<sup>every</sup>  $i$ , what's the complexity?  
naive:  $O(n^2)$

# Reusing computation



**Compute:**

$$P(X_i | x_0, x_{n+1})$$

$$P(X_4, x_0, x_5) = \sum_{x_1, x_2, x_3} P(x_0) P(x_1 | x_0) P(x_2 | x_1) \dots$$

$$= P(x_0) \sum_{x_2, x_3} P(x_3 | x_2) \dots \underbrace{\sum_{x_1} P(x_1 | x_0) P(x_2 | x_1)}_{g_1(x_2)}$$

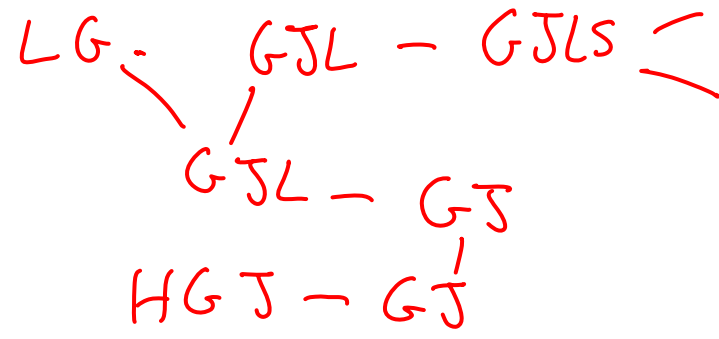
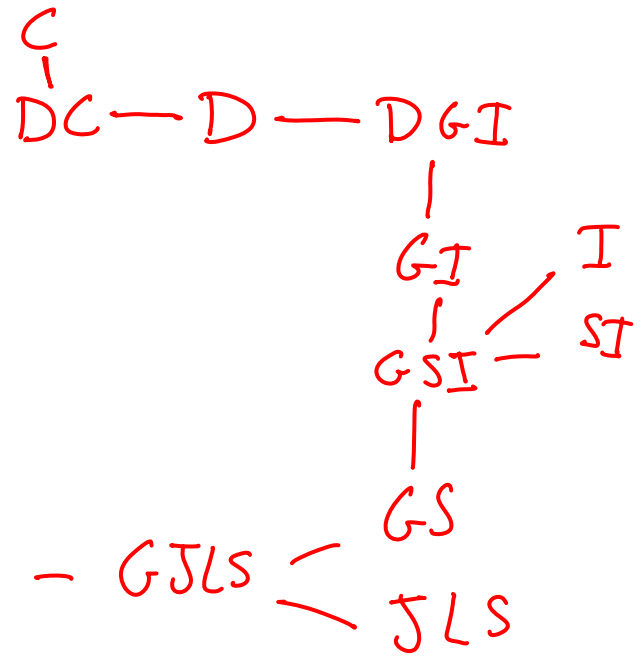
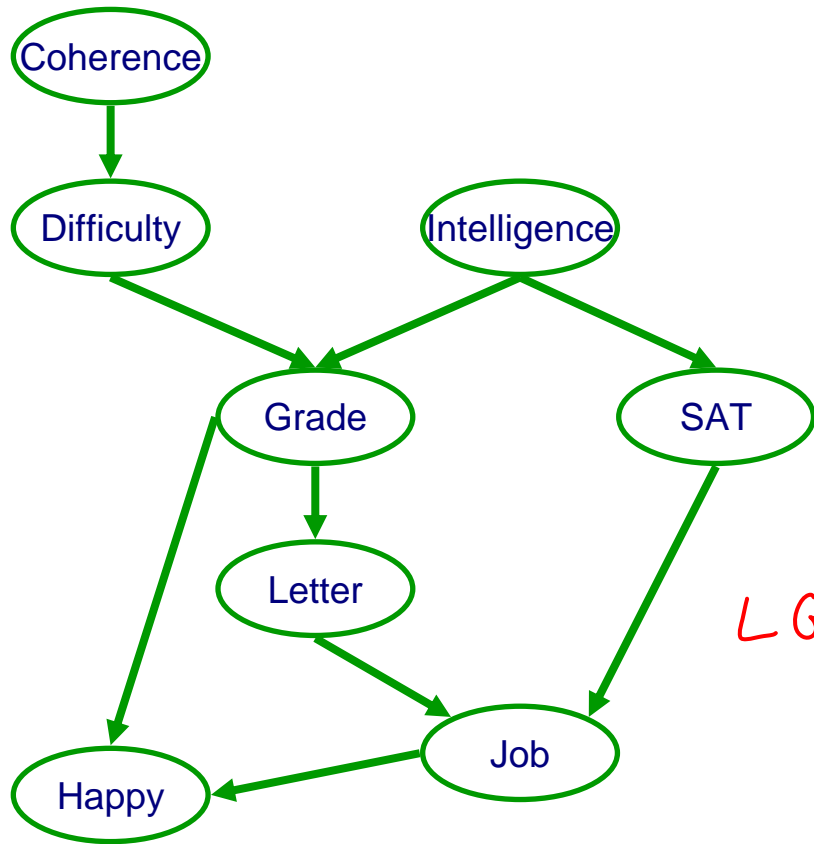
---


$$P(X_3, x_0, x_5) = \sum_{x_1, x_2, x_4} P(x_0) P(x_1 | x_0) \dots = \sum_{x_2, x_4} P(x_3 | x_2) \dots \underbrace{\sum_{x_1} P(x_1 | x_0) P(x_2 | x_1)}_{g_1(x_2)}$$



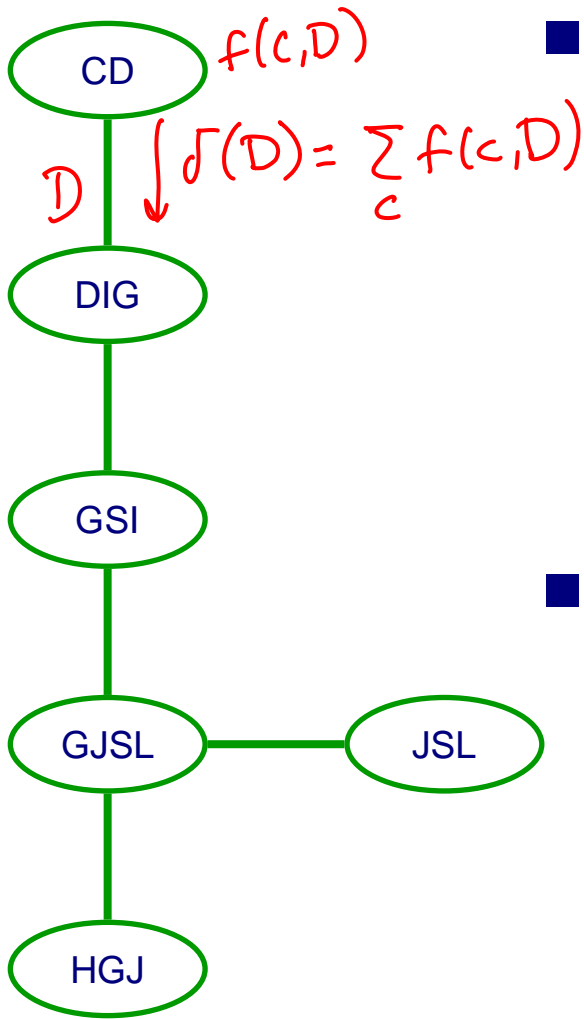
# Factors generated by VE

$P(C)$   $P(D|C)$   $P(I)$   
 $P(G|DI)$   $P(S|I)$   
 $P(L|G)$   $P(J|L,S)$   
 $P(H|G,J)$



**Elimination order:**  
~~{C,D,I,S,L,H,J,G}~~

# Cluster graph for VE



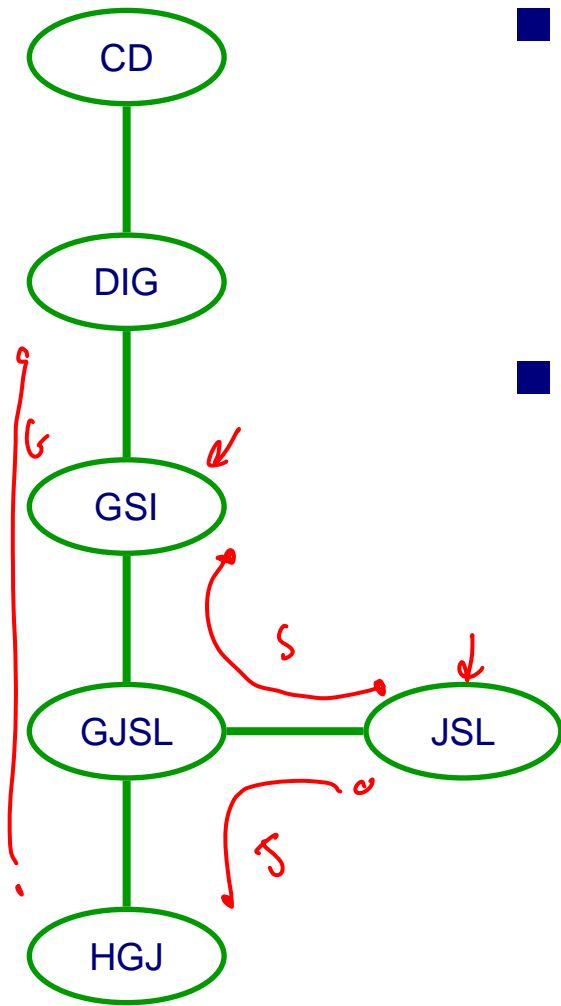
## ■ VE generates cluster tree!

- One clique for each factor used/generated
- Edge  $i - j$ , if  $f_i$  used to generate  $f_j$
- “Message” from  $i$  to  $j$  generated when marginalizing a variable from  $f_i$
- Tree because factors only used once

## ■ Proposition:

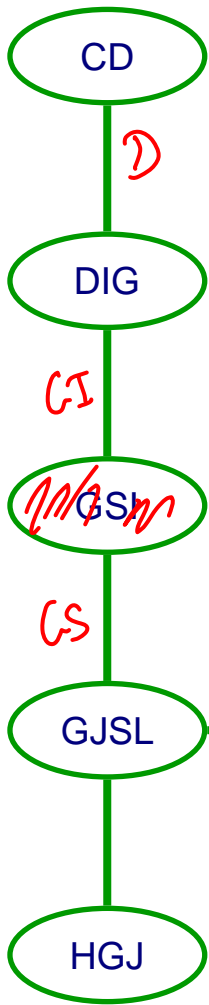
- “Message”  $\delta_{ij}$  from  $i$  to  $j$
- Scope $[\delta_{ij}] \subseteq \mathbf{S}_{ij}$

# Running intersection property



- **Running intersection property (RIP)**
  - Cluster tree satisfies RIP if whenever  $X \in \mathbf{C}_i$  and  $X \in \mathbf{C}_j$  then  $X$  is in every cluster in the (unique) path from  $\mathbf{C}_i$  to  $\mathbf{C}_j$
- **Theorem:**
  - Cluster tree generated by VE satisfies RIP

# Clique tree & Independencies



- **Clique tree (or Junction tree)**

- A cluster tree that satisfies the RIP

- **Theorem:**

- Given some BN with structure  $G$  and factors  $F$

- For a clique tree  $T$  for  $F$  consider  $C_i - C_j$  with separator  $S_{ij}$ :

- $X$  – any set of vars in  $C_i$  side of the tree (C ⊥ L | GS)

- $Y$  – any set of vars in  $C_j$  side of the tree (C ⊥ L | D)

- Then,  $(X \perp Y | S_{ij})$  in BN

- Furthermore,  $I(T) \subseteq I(G)$

# Variable elimination in a clique tree 1

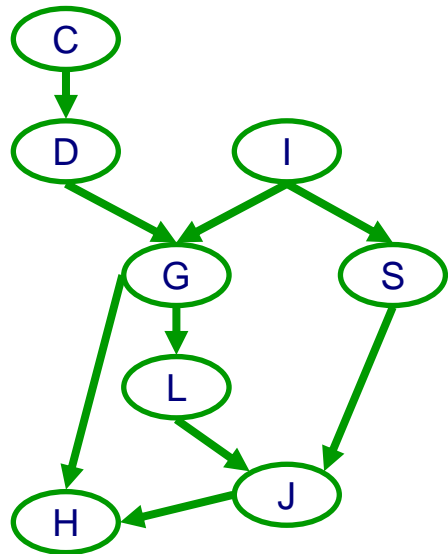
$C_1$ : CD

$C_2$ : DIG

$C_3$ : GSI

$C_4$ : GJSL

$C_5$ : HGJ



## ■ Clique tree for a BN

- Each CPT assigned to a clique
- Initial potential  $\pi_0(\mathbf{C}_i)$  is product of CPTs

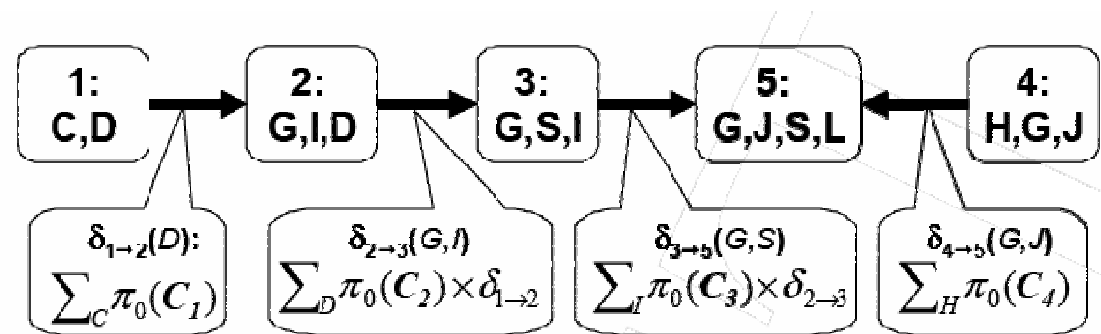
# Variable elimination in a clique tree 2



## ■ VE in clique tree to compute $P(X_i)$

- Pick a root (any node containing  $X_i$ )
- Send messages recursively from leaves to root
  - Multiply incoming messages with initial potential
  - Marginalize vars that are not in separator
- Clique *ready* if received messages from all neighbors

# Beliefs from messages



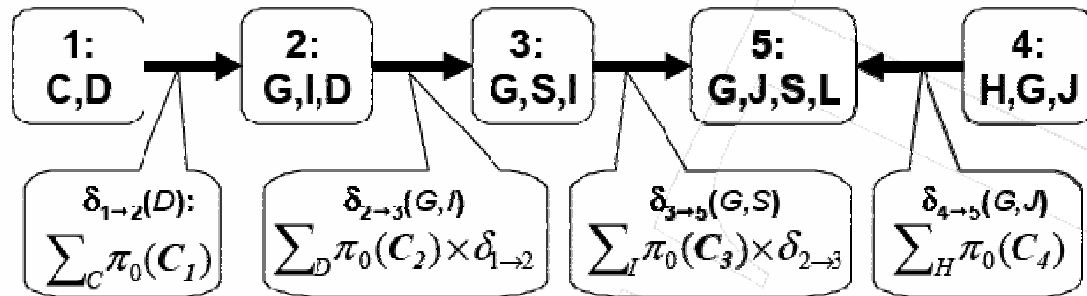
## ■ Theorem: When clique $C_i$ is ready

- Receive messages from all neighbors
- Belief  $\pi_i(C_i)$  is product of initial factor with messages:

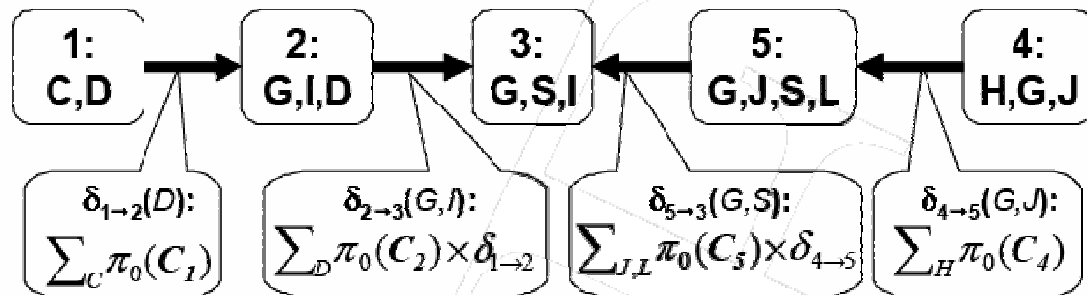
# Choice of root

- Message does not depend on root!!!

Root: node 5

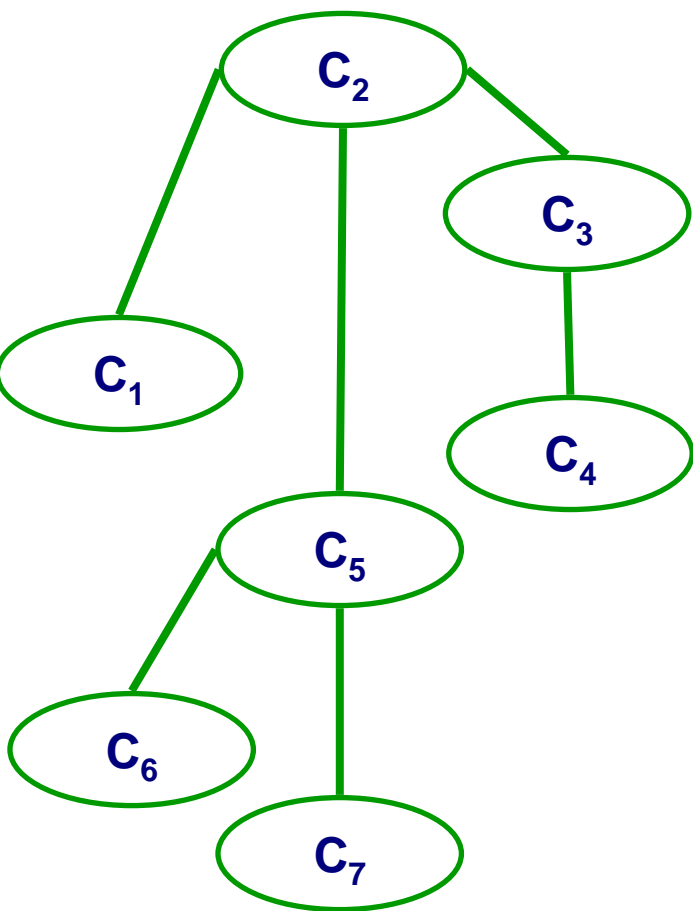


Root: node 3



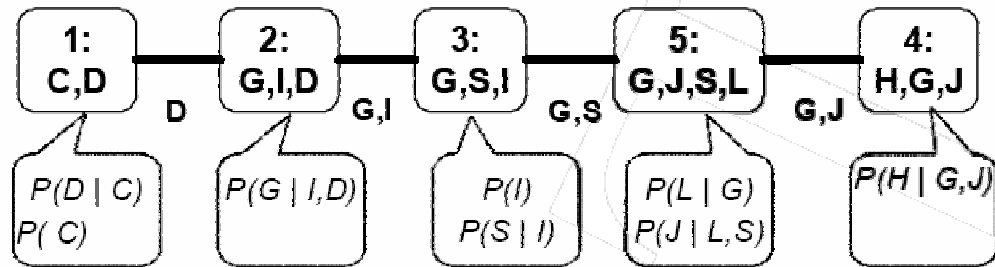
“Cache” computation: Obtain belief for all roots in linear time!!

# Shafer-Shenoy Algorithm (a.k.a. VE in clique tree for all roots)



- Clique  $C_i$  ready to transmit to neighbor  $C_j$  if received messages from all neighbors but  $j$ 
  - Leaves are always ready to transmit
- While  $\exists C_i$  ready to transmit to  $C_j$ 
  - Send message  $\delta_{i \rightarrow j}$
- Complexity: Linear in # cliques
  - One message sent each direction in each edge
- **Corollary:** At convergence
  - Every clique has correct belief

# Calibrated Clique tree



- Initially, neighboring nodes don't agree on "distribution" over separators
- **Calibrated clique tree:**
  - At convergence, tree is *calibrated*
  - Neighboring nodes agree on distribution over separator

# Message passing with division



$C_1$ : CD

$C_2$ : DIG

$C_3$ : GSI

$C_4$ : GJSL

$C_5$ : HGJ

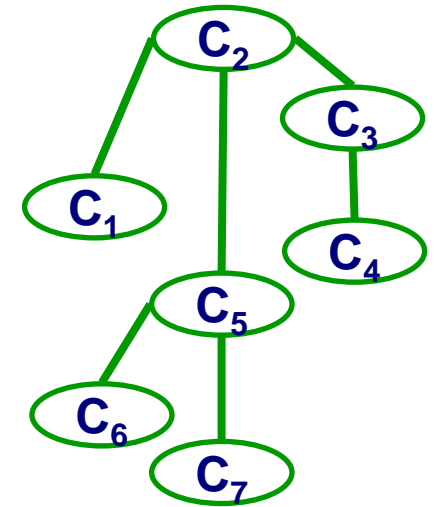
- Computing messages by multiplication:

- Computing messages by division:

# Lauritzen-Spiegelhalter Algorithm (a.k.a. belief propagation)

Simplified description  
see reading for details

- Initialize all separator potentials to 1
  - $\mu_{ij} \leftarrow 1$
- All messages ready to transmit
- While  $\exists \delta_{i \rightarrow j}$  ready to transmit
  - $\mu_{ij}' \leftarrow$
  - If  $\mu_{ij}' \neq \mu_{ij}$ 
    - $\delta_{i \rightarrow j} \leftarrow$
    - $\pi_j \leftarrow \pi_j \times \delta_{i \rightarrow j}$
    - $\mu_{ij} \leftarrow \mu_{ij}'$
    - $\forall$  neighbors  $k$  of  $j$ ,  $k \neq i$ ,  $\delta_{j \rightarrow k}$  ready to transmit
- Complexity: Linear in # cliques
  - for the “right” schedule over edges (leaves to root, the root to leaves)
- **Corollary:** At convergence, every clique has correct belief





# Clique tree invariant

- **Clique tree potential:**
  - Product of clique potentials divided by separators potentials
  
- **Clique tree invariant:**
  - $P(\mathbf{X}) = \pi_T$

# Belief propagation and clique tree invariant

- **Theorem:** Invariant is maintained by BP algorithm!
- BP reparameterizes potentials and messages
  - At convergence, potentials and messages are marginal distributions

# Subtree correctness

- **Informed message** from  $i$  to  $j$ , if all messages into  $i$  (other than from  $j$ ) are informed
  - Recursive definition (leaves always send informed messages)
- **Informed subtree:**
  - All incoming messages informed
- **Theorem:**
  - Potential of connected informed subtree  $T'$  is marginal over  $\text{scope}[T']$
- **Corollary:**
  - At convergence, clique tree is *calibrated*
    - $\pi_i = P(\text{scope}[\pi_i])$
    - $\mu_{ij} = P(\text{scope}[\mu_{ij}])$

# Answering queries with clique trees

- Query within clique
- Incremental updates – Observing evidence  $Z=z$ 
  - Multiply some clique by indicator  $\mathbf{1}(Z=z)$
- Query outside clique
  - Use variable elimination!

# Constructing a clique tree from VE

- Select elimination order

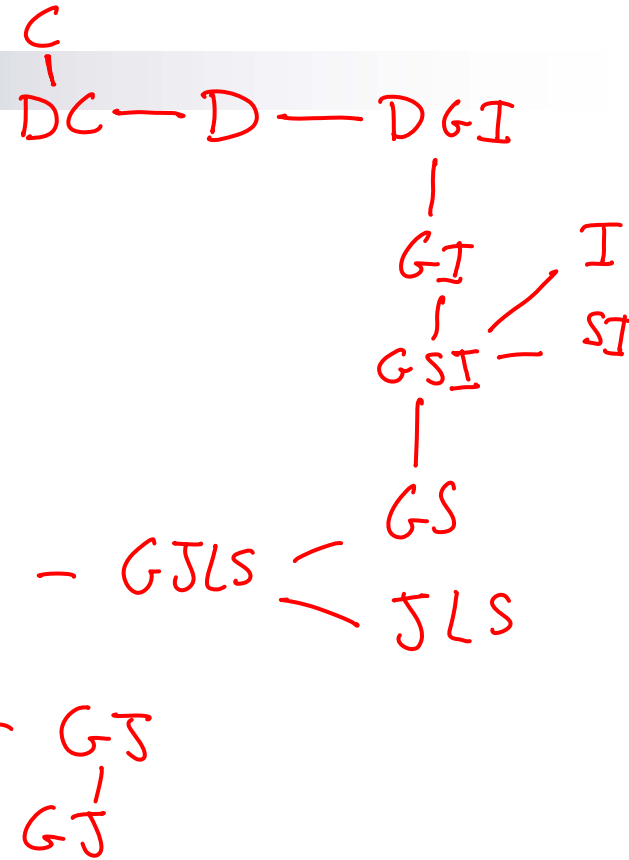
↖

- Connect factors that would be generated if you run VE with order

↖

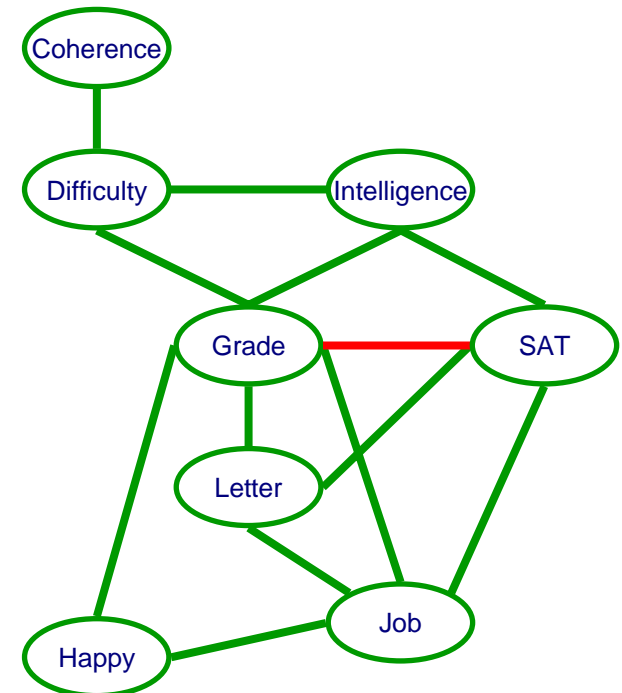
- Simplify!

- Eliminate factor that is subset of neighbor



# Find clique tree from chordal graph

- Triangulate moralized graph to obtain chordal graph
- Find maximal cliques
  - NP-complete in general
  - Easy for chordal graphs
  - Max-cardinality search from last lecture
- Generate weighted graph over cliques
  - Edge weights  $(i,j)$  is separator size  $- |C_i \cap C_j|$
- Maximum spanning tree finds clique tree satisfying RIP!!!



# Clique trees versus VE

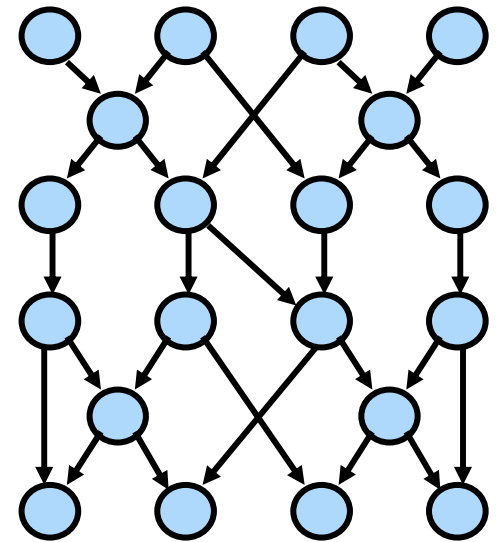
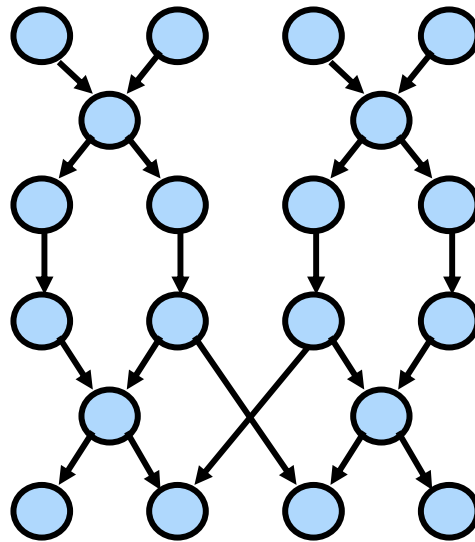
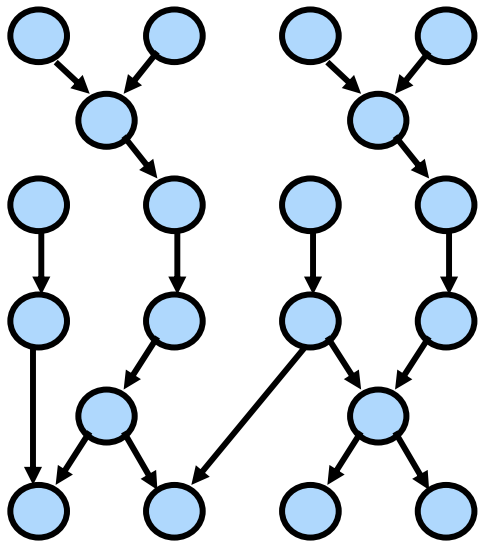


- Clique tree advantages
  - Multi-query settings
  - Incremental updates
  - Pre-computation makes complexity explicit
  
- Clique tree disadvantages
  - Space requirements – no factors are “deleted”
  - Slower for single query
  - Local structure in factors may be lost when they are multiplied together into initial clique potential

# Clique tree summary

- Solve marginal queries for all variables in only twice the cost of query for one variable
- Cliques correspond to maximal cliques in induced graph
- Two message passing approaches
  - VE (the one that multiplies messages)
  - BP (the one that divides by old message)
- Clique tree invariant
  - Clique tree potential is always the same
  - We are only reparameterizing clique potentials
- Constructing clique tree for a BN
  - from elimination order
  - from triangulated (chordal) graph
- Running time (only) exponential in size of largest clique
  - Solve **exactly** problems with thousands (or millions, or more) of variables, and cliques with tens of nodes (or less)

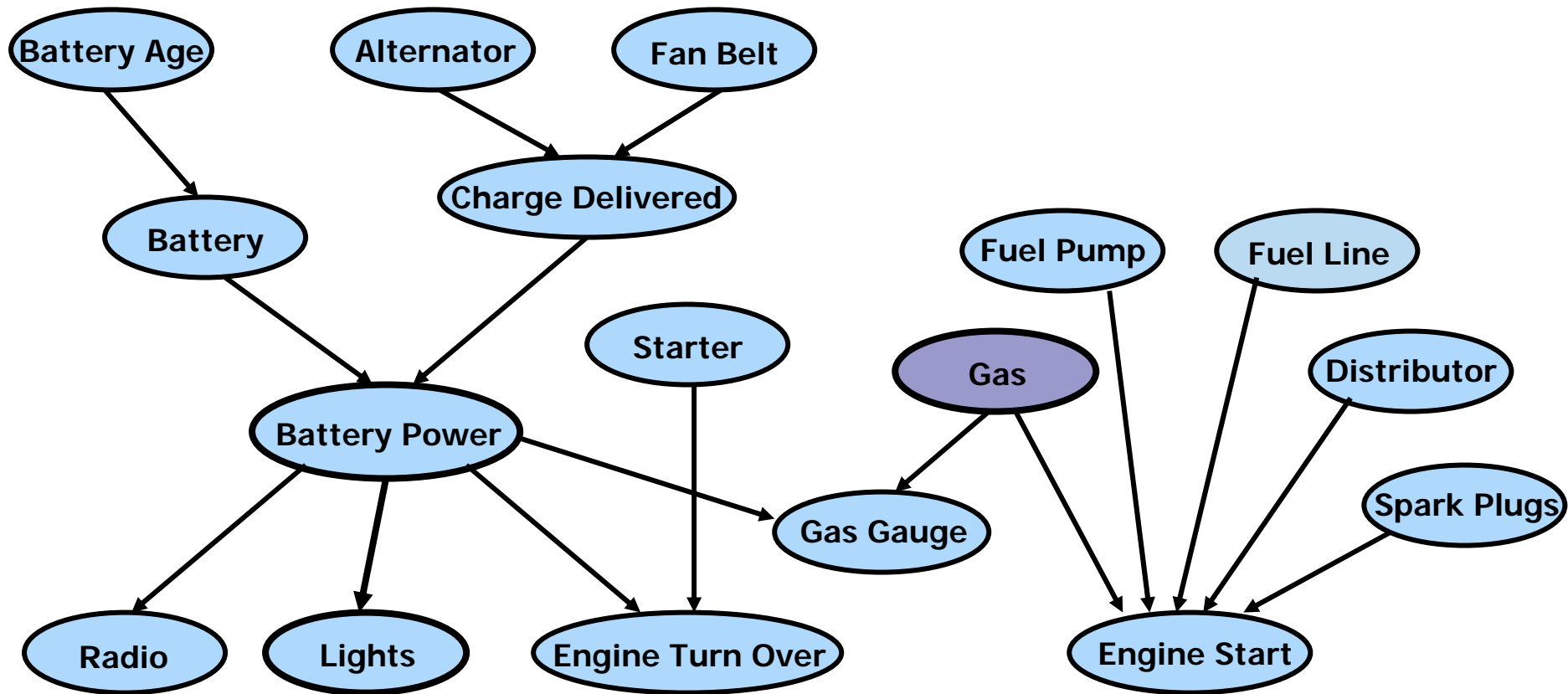
# Global Structure: Treewidth $w$



$O(n \exp(w))$

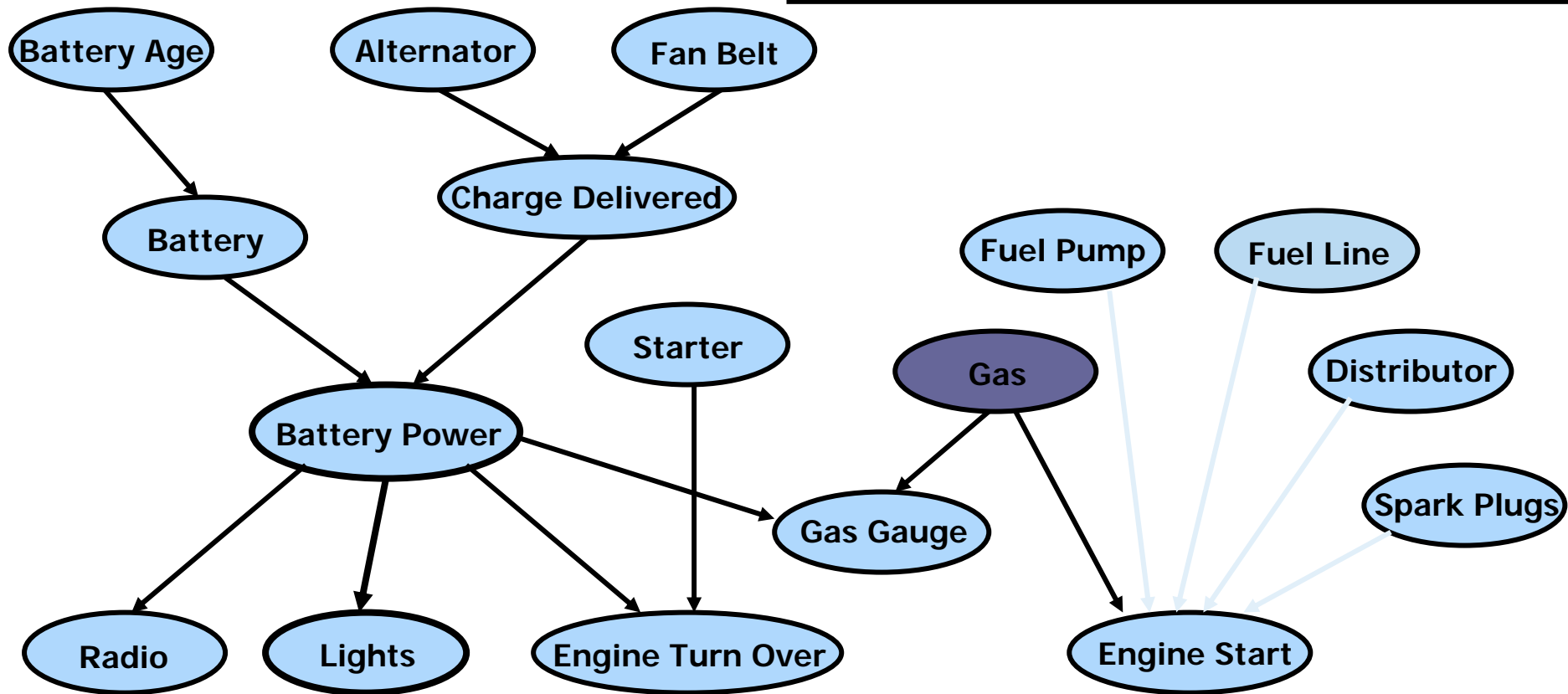


# Local Structure 1: Context specific independence



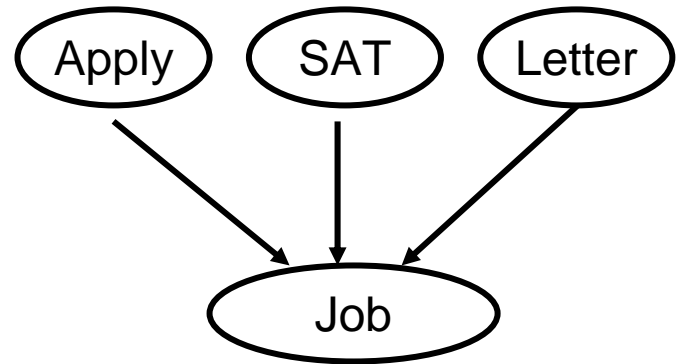
# Local Structure 1: Context specific independence

Context Specific Independence (CSI)  
After observing a variable, some vars become independent

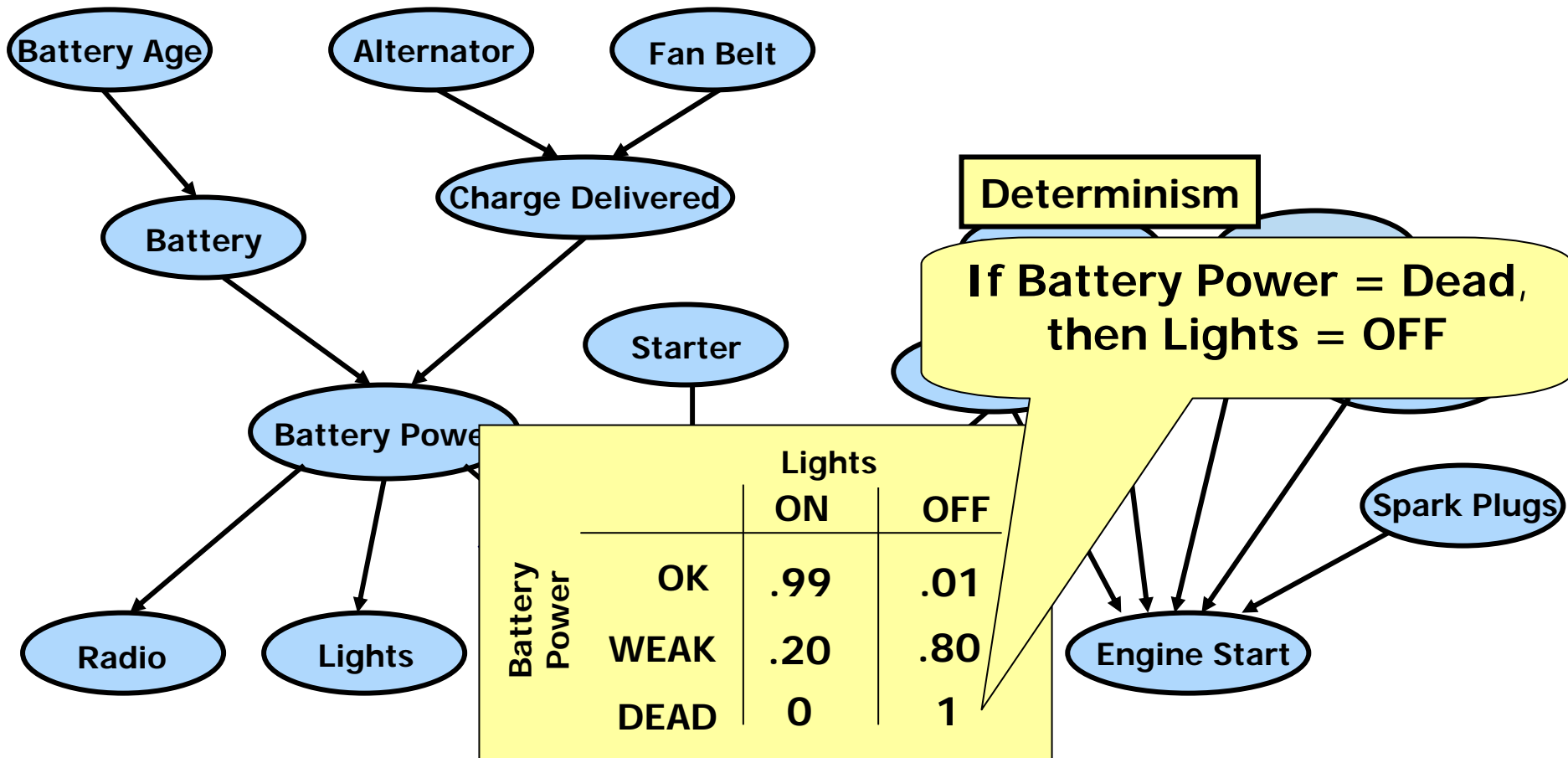


# CSI example: Tree CPD

- Represent  $P(X_i | \mathbf{Pa}_{X_i})$  using a decision tree
  - Path to leaf is an assignment to (a subset of)  $\mathbf{Pa}_{X_i}$
  - Leaves are distributions over  $X_i$  given assignment of  $\mathbf{Pa}_{X_i}$  on path to leaf
- **Interpretation of leaf:**
  - For specific assignment of  $\mathbf{Pa}_{X_i}$  on path to this leaf –  $X_i$  is independent of other parents
- Representation can be exponentially smaller than equivalent table



# Local Structure 2: Determinism



# Today's Models ...

- **Often characterized by:**

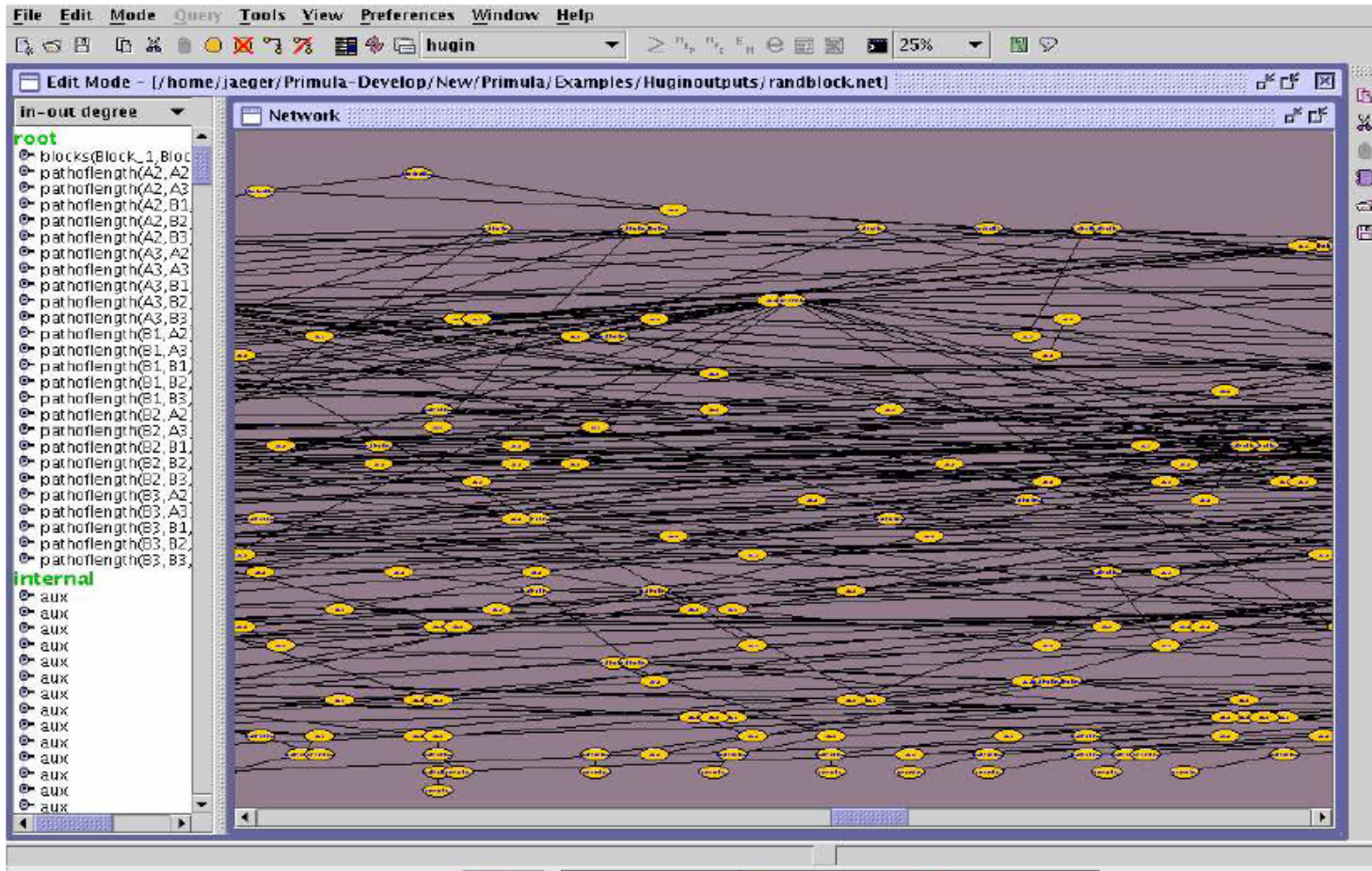
- Richness in local structure (determinism, CSI)
- Massiveness in size (10,000's variables)
- High connectivity (treewidth)

- **Enabled by:**

- High level modeling tools: relational, first order
- Advances in machine learning
- New application areas (synthesis):
  - Bioinformatics (e.g. linkage analysis)
  - Sensor networks

- **Exploiting local structure a must!**

# Exact inference in large models is possible...



- BN from a relational model