Jin Gu
Department of Automation, Tsinghua University
Email: jgu@tsinghua.edu.cn
Phone: (010) 62794294-866

Chapter 8

Inference as Optimization:
Cluster Graph & Belief Propagation

2021 Fall

Jin Gu (古槿)

Outlines

- Variable Elimination (消元法)
  - Simple case: linear chain Bayesian networks
  - VE in complex graphs
  - Inferences in HMMs and linear-chain CRFs
- Exact Inference: Clique Tree
  - Cluster graph and clique tree
  - Message passing: sum product
  - Message passing: belief update
  - Constructing clique tree

Outlines

- From clique tree to loopy cluster graph
- Bethe cluster graph or cluster graph
- Belief propagation as variational inference
- Extensions of belief propagation
  - Generalized belief propagation
  - Convex belief propagation
  - Expectation propagation
  - ......

Symbols

Φ	A set of factors over a set of variables χ
ψ	Potential
ϕ	Factor
τ	Auxiliary factor during variable elimination. Also called *"message"* .
$\mathcal{U}$	Cluster graph
C_i	A subset $C_i \subseteq \chi$, associated with a cluster/clique node in cluster/clique graph.
$\alpha(\phi)$	Assignment of a factor to a cluster, for example: $Scope[\phi] \subseteq C_i$
$S_{i,j} \subseteq C_i \cap C_j$	Sepset associated with edge between C_i and C_j
$\mathcal{T}$	Cluster tree
$\delta_{i \to j}(X)$	Message about $X \subseteq \chi$ send from C_i to C_j
$\beta_i(C_i)$	Belief of clique C_i
$\mu_{i,j}(S_{i,j})$	Belief of sepset $S_{i,j}$

Chapter 8 Cluster Graph & Belief Propagation

Textbook1

Chapter 9.2-9.3 Variable Elimination
Chapter 10.1-10.3 Clique Tree & Message Passing
Chapter 11.2 Exact Inference as Optimization
Chapter 11.3.1-11.3.3 General Cluster Graph & Message Passing
Chapter 11.3.5 Construction of Cluster Graph
***Chapter 11.3.5** Varational Analysis

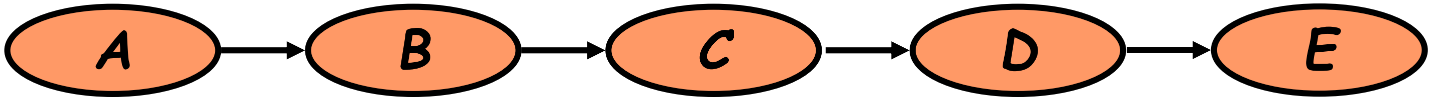
Textbook2

Chapter 20 Exact Inference for Graphical Models
Chapter 22 More Variational Inference

Other references

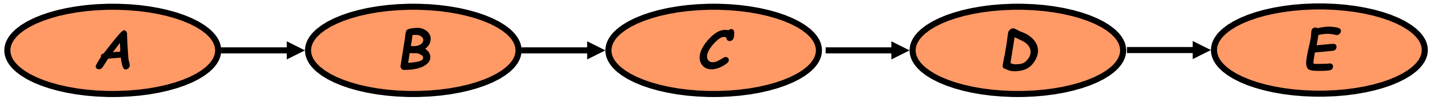
Wainwright MJ, Michael IJ. **Graphical models, exponential families, and variational inference**. *Foundations and Trends® in Machine Learning*, 1(1-2):1-305.

Simple Case: VE in Chains



$$P(e) = \sum_{d}\sum_{c}\sum_{b}\sum_{a} P(a,b,c,d,e)$$

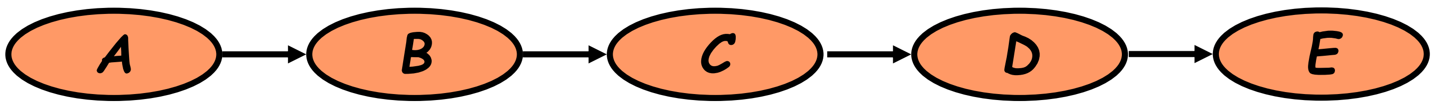
Simple Case: VE in Chains



- By the chain rule for probabilities:

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a,b,c,d,e)$$
$$= \sum_d \sum_c \sum_b \sum_a P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d)$$

Simple Case: VE in Chains

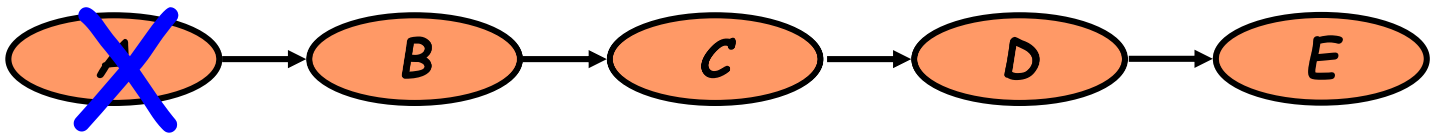


- Rearranging terms ...

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a)P(b \mid a)\overbrace{P(c \mid b)P(d \mid c)P(e \mid d)}$$

$$= \sum_d \sum_c \sum_b P(c \mid b)P(d \mid c)P(e \mid d)\sum_a P(a)P(b \mid a)$$

Simple Case: VE in Chains

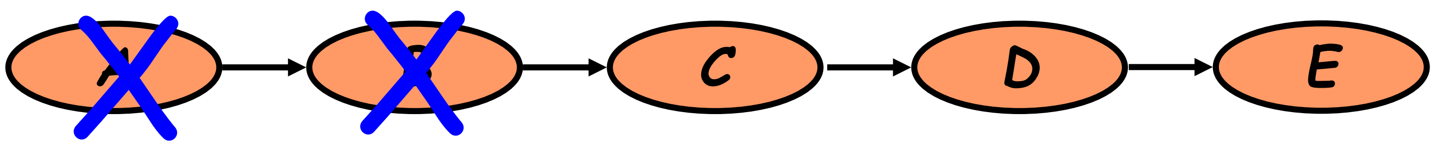


- Perform the innermost summation

$$P(e) = \sum_d \sum_c \sum_b P(c \mid b) P(d \mid c) P(e \mid d) \sum_a P(a) P(b \mid a)$$

$$= \sum_d \sum_c \sum_b P(c \mid b) P(d \mid c) P(e \mid d) p(b)$$

Simple Case: VE in Chains



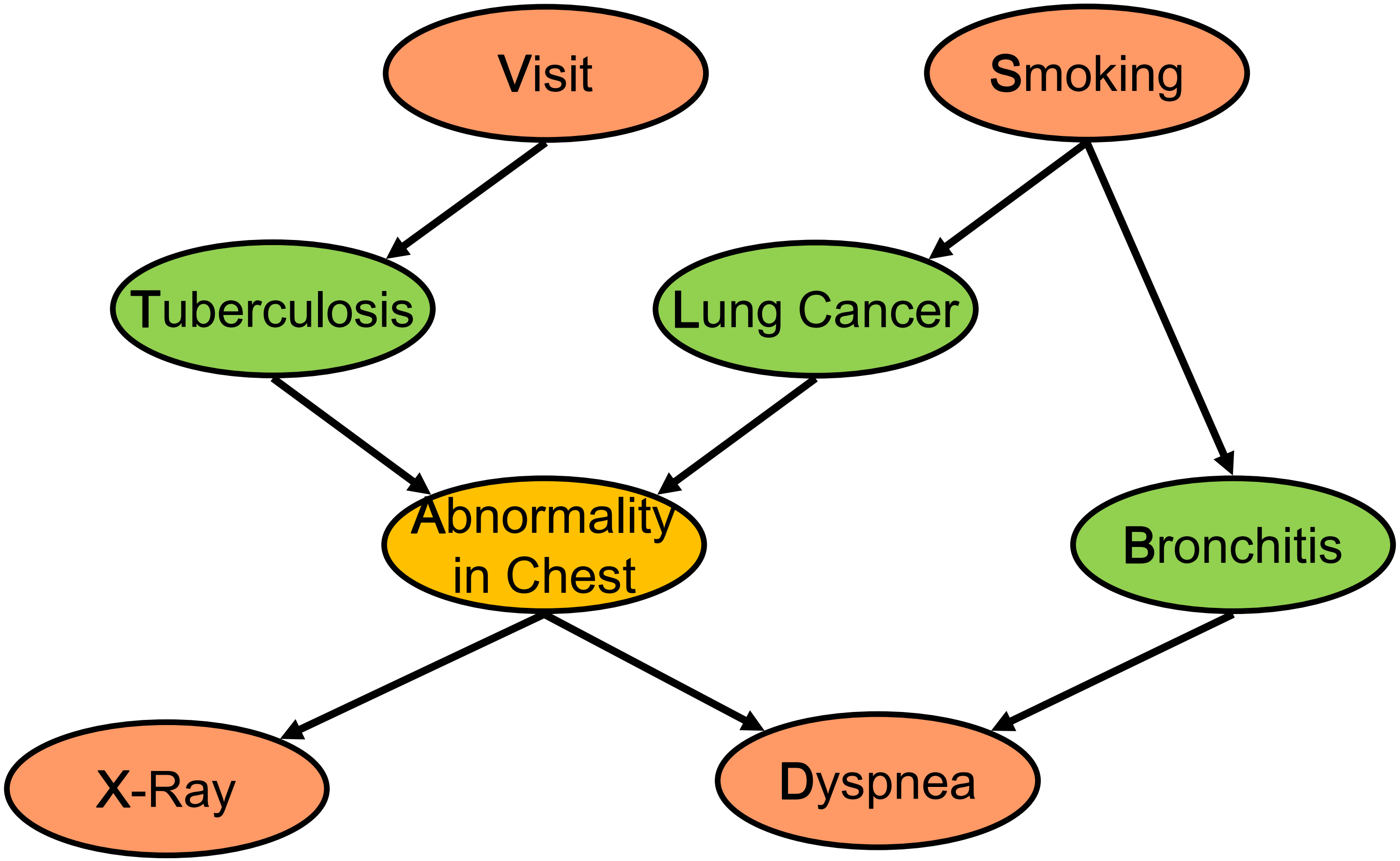
- Rearrange and then sum again

$$P(e) = \sum_d \sum_c \sum_b P(c \mid b) P(d \mid c) P(e \mid d) p(b)$$

$$= \sum_d \sum_c P(d \mid c) P(e \mid d) \sum_b P(c \mid b) p(b)$$

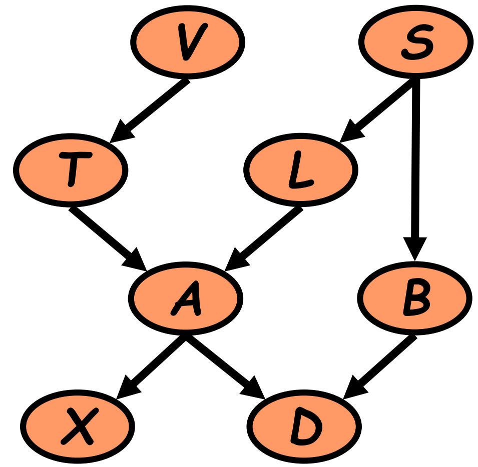
$$= \sum_d \sum_c P(d \mid c) P(e \mid d) p(c)$$

General Bayesian Networks



Example: a simplified lung disease diagnostic network

- We want to compute $P(D)$
- Need to eliminate: V,S,X,T,L,A,B

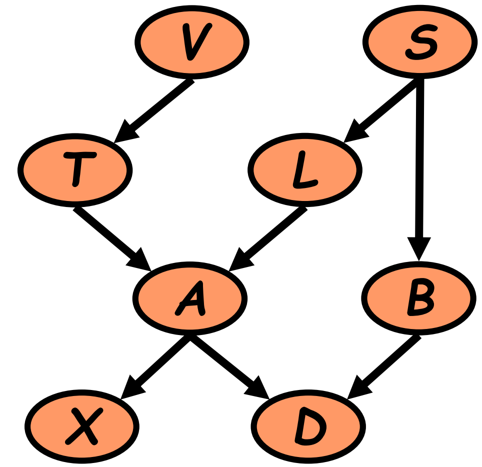


Initial factors

$$P(V)P(S)P(T\,|\,V)P(L\,|\,S)P(B\,|\,S)P(A\,|\,T,L)P(X\,|\,A)P(D\,|\,A,B)$$

$$\phi_V(V)\phi_S(S)\phi_T(T,V)\phi_L(L,S)\phi_B(B,S)\phi_A(A,T.L)\phi_X(X,A)\phi_D(D,A,B)$$

- We want to compute *P(D)*
- Need to eliminate: *V,S,X,T,L,A,B*



Current factors

$$\phi_V(V)\phi_S(S)\phi_T(T,V)\phi_L(L,S)\phi_B(B,S)\phi_A(A,T.L)\phi_X(X,A)\phi_D(D,A,B)$$

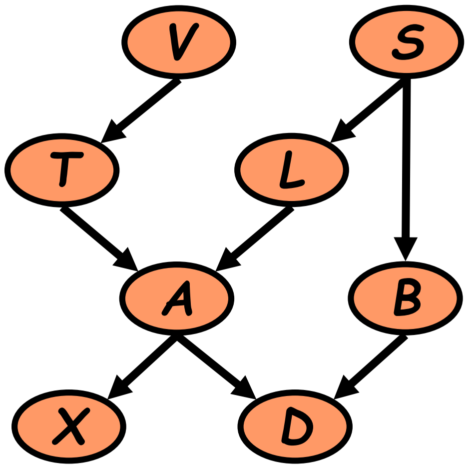
Eliminate: *V*

Compute: $\tau_1(T) = \sum_V \phi_V(V)\phi_T(T,V)$

$$\tau_1(T)\phi_S(S)\phi_L(L,S)\phi_B(B,S)\phi_A(A,T.L)\phi_X(X,A)\phi_D(D,A,B)$$

Note: $\tau_1(T) = P(T)$

- We want to compute *P(D)*
- Need to eliminate: *S,X,T,L,A,B*



Current factors

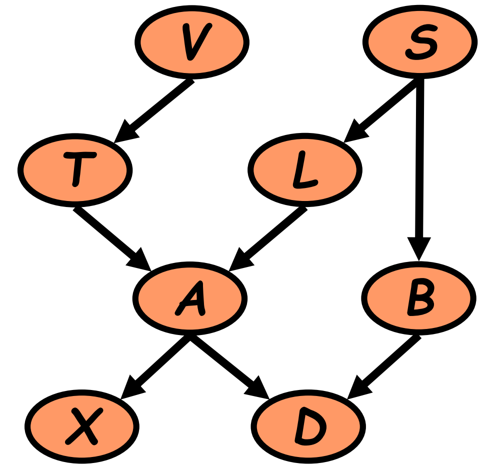
$$\tau_1(T)\underline{\phi_S(S)}\,\underline{\phi_L(L,S)}\,\underline{\phi_B(B,S)}\phi_A(A,T.L)\phi_X(X,A)\phi_D(D,A,B)$$

Eliminate: *S*

Compute: $$\tau_2(L,B)=\sum_V \phi_S(S)\phi_L(L,S)\phi_B(B,S)$$

$$\tau_1(T)\tau_2(B,L)\phi_A(A,T.L)\phi_X(X,A)\phi_D(D,A,B)$$

- We want to compute *P(D)*
- Need to eliminate: *X,T,L,A,B*



Current factors

$$\tau_1(T)\tau_2(B,L)\phi_A(A,T.L)\underline{\phi_X(X,A)}\phi_D(D,A,B)$$

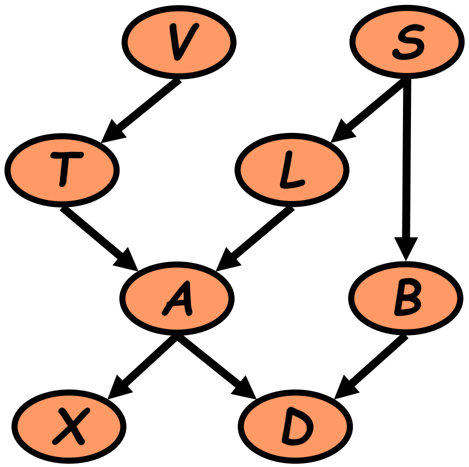
Eliminate: *X*

Compute: $\tau_3(A) = \sum_X \phi_X(X,A)$

$$\Rightarrow \quad \tau_1(T)\tau_2(B,L)\phi_A(A,T.L)\tau_3(A)\phi_D(D,A,B)$$

Note: $\tau_3(A) = 1$ for all values of A !!

- We want to compute *P(D)*
- Need to eliminate: *T,L,A,B*



Current factors

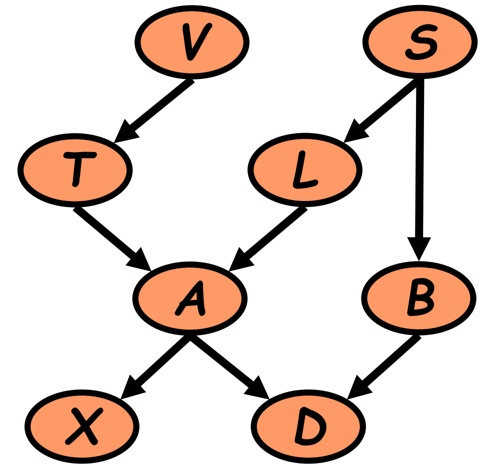
$$\underline{\tau_1(T)}\tau_2(B,L)\underline{\phi_A(A,T.L)}\tau_3(A)\phi_D(D,A,B)$$

Eliminate: T

Compute: $\tau_4(A,L) = \sum_T \tau_1(T)\phi_A(A,T,L)$

$$\Rightarrow \quad \tau_4(A,L)\tau_2(B,L)\tau_3(A)\phi_D(D,A,B)$$

- We want to compute $P(D)$
- Need to eliminate: L,A,B



Current factors

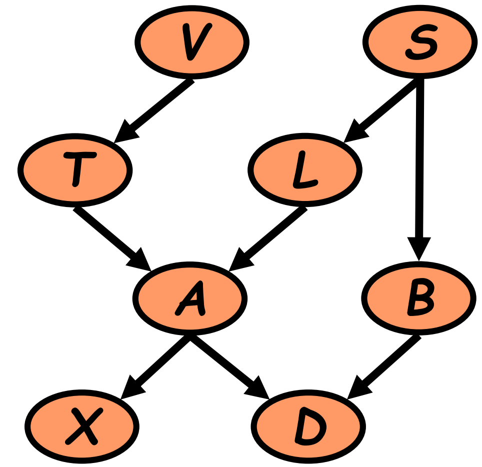
$$\underline{\tau_4(A,L)}\,\underline{\tau_2(B,L)}\tau_3(A)\phi_D(D,A,B)$$

Eliminate: L

Compute: $$\tau_5(A,B) = \sum_L \tau_4(A,L)\tau_2(B,L)$$

$$\tau_5(A,B)\tau_3(A)\phi_D(D,A,B)$$

- We want to compute $P(D)$
- Need to eliminate: A,B



Current factors

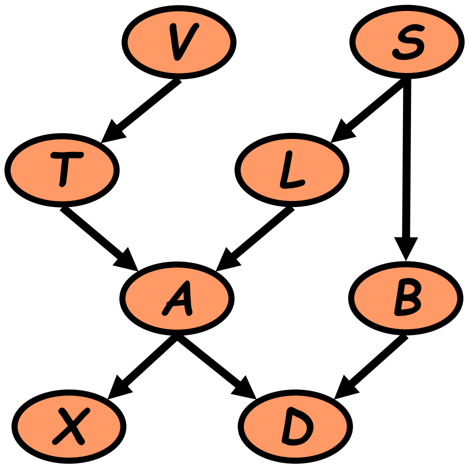
$$\tau_5(A,B)\tau_3(A)\phi_D(D,A,B)$$

Eliminate: *A*

Compute: $\tau_6(B,D) = \sum_A \tau_5(A,B)\tau_3(A)\phi_D(D,A,B)$

$\Rightarrow$ $\tau_6(B,\ D)$

- We want to compute $P(D)$
- Need to eliminate: B



Current factors

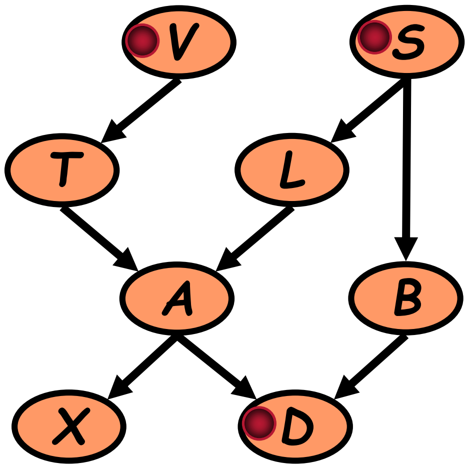
$\underline{\tau_6(B, D)}$

Eliminate: B

Compute: $\tau_7(D) = \sum_B \tau_6(B, D)$

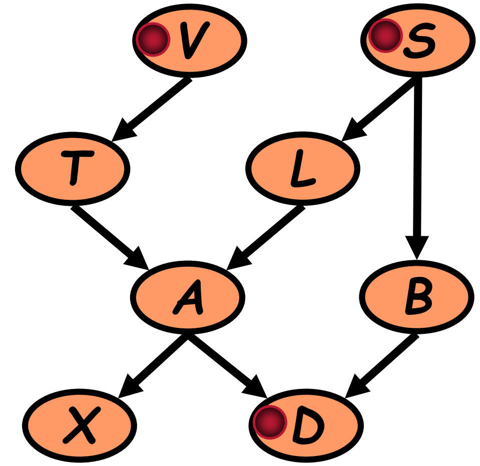
Note: $\tau_7(D)$ is $P(D)$

Dealing with Evidence



- How do we deal with evidence?
- Suppose get evidence $V = v, S = s, D = d$
- We want to compute $P(L, V = v, S = s, D = d)$

Dealing with Evidence

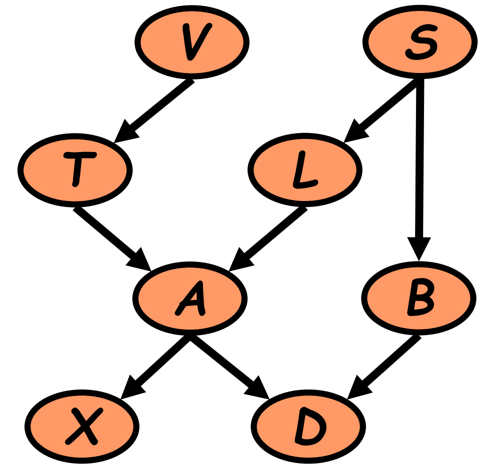


- Compute $P(L, V = v, S = s, D = d)$
- Initial factors, after setting evidence:

$$\phi_V(v)\phi_S(s)\phi_T(T, v)\phi_L(L, s)\phi_B(B, s)\phi_A(A, T, L)\phi_X(X, A)\phi_D(d, A, B)$$

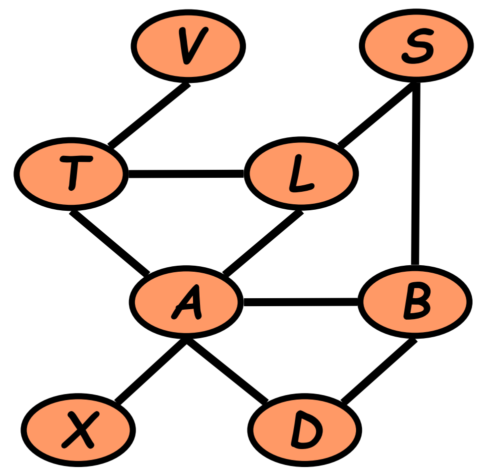
$$\tilde{\phi}_V()\tilde{\phi}_S()\tilde{\phi}_T(T)\tilde{\phi}_L(L)\tilde{\phi}_B(B)\phi_A(A, T, L)\phi_X(X, A)\tilde{\phi}_D(A, B)$$

Induced Graph in VE

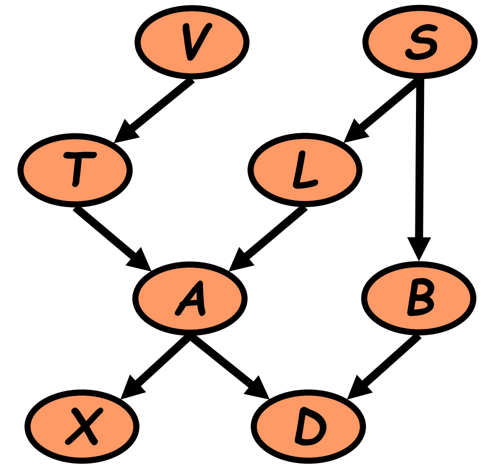


- Want to compute $P(L)$

- Step 1: Moralizing

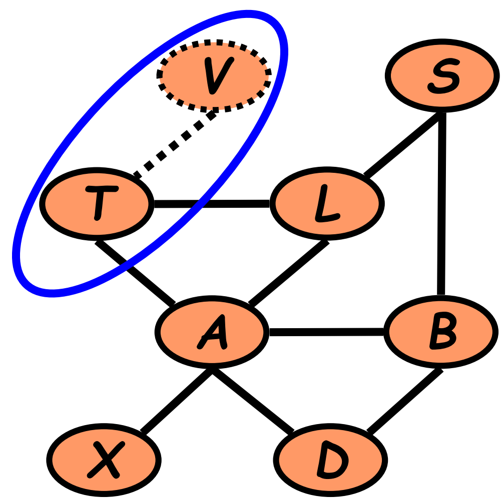


Induced Graph in VE

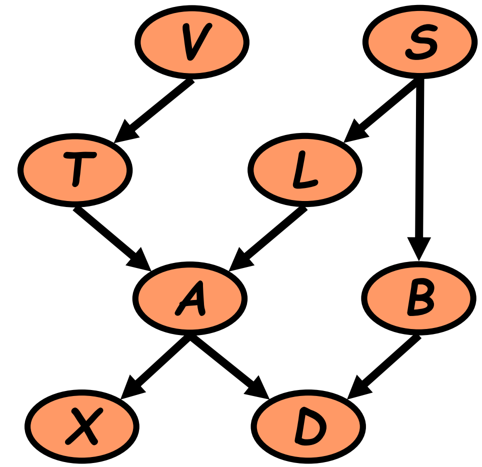


- Want to compute $P(L)$
- Moralizing
- Eliminating V

$$\tau_1(T) = \sum_V \phi_V(V)\phi_T(T,V)$$

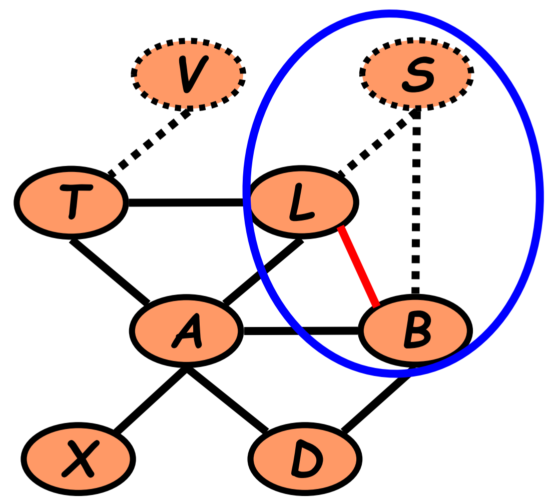


Induced Graph in VE



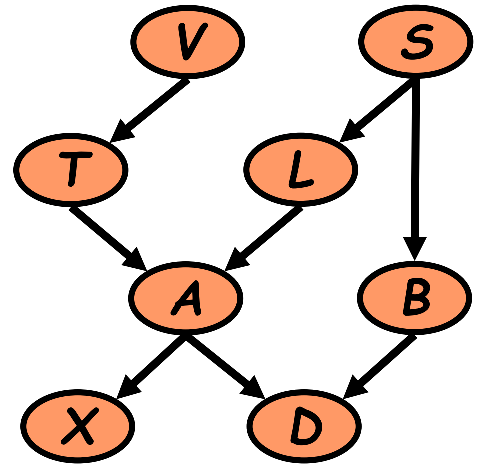
- Want to compute *P(L)*

- Moralizing
- Eliminating *V*
- Eliminating *S*

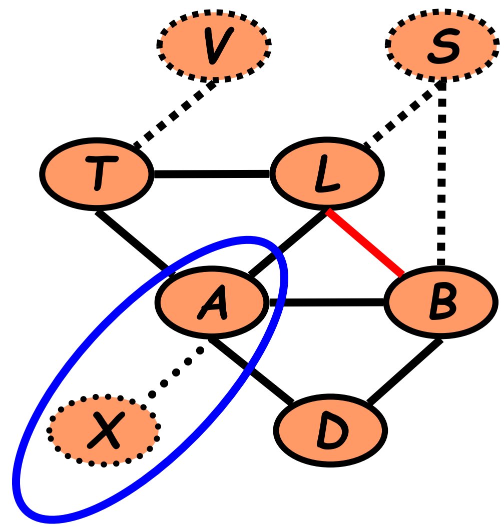


$$\tau_2(L,B) = \sum_V \phi_S(S)\phi_L(L,S)\phi_B(B,S)$$

Induced Graph in VE



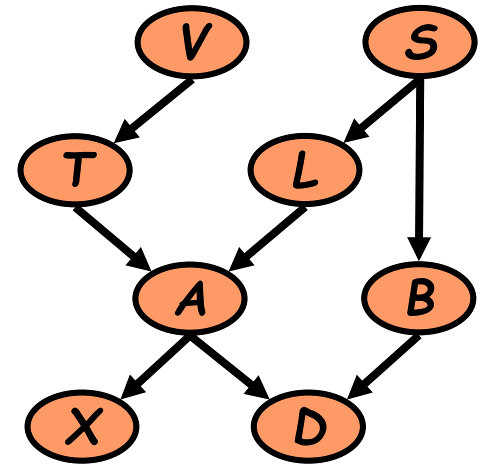
- Want to compute $P(L)$
- Moralizing
- Eliminating V
- Eliminating S
- Eliminating X



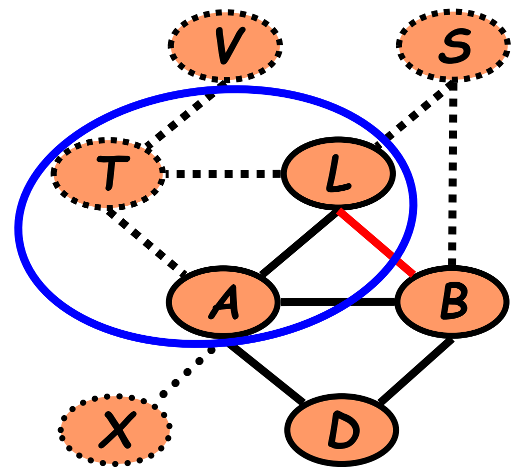
$$\tau_3(A) = \sum_X \phi_X(X, A)$$

Induced Graph in VE

- Want to compute $P(D)$

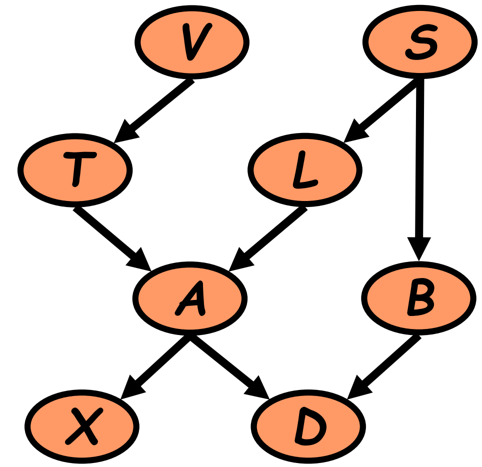


- Moralizing
- Eliminating V
- Eliminating S
- Eliminating X
- Eliminating T



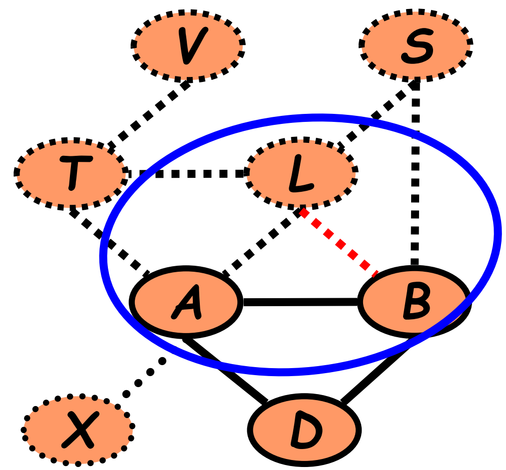
$$\tau_5(A,B) = \sum_T \tau_4(A,L)\tau_2(B,L)$$

Induced Graph in VE

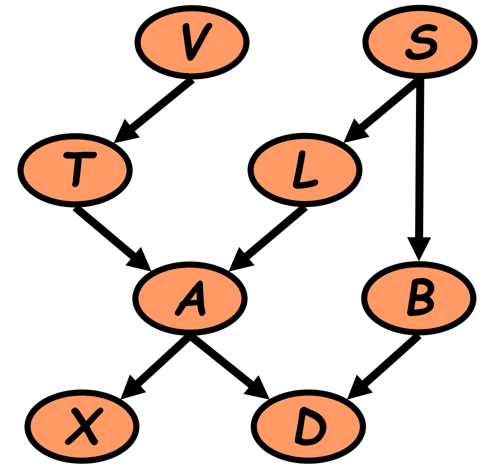


- Want to compute $P(D)$
- Moralizing
- Eliminating V
- Eliminating S
- Eliminating X
- Eliminating T
- Eliminating L

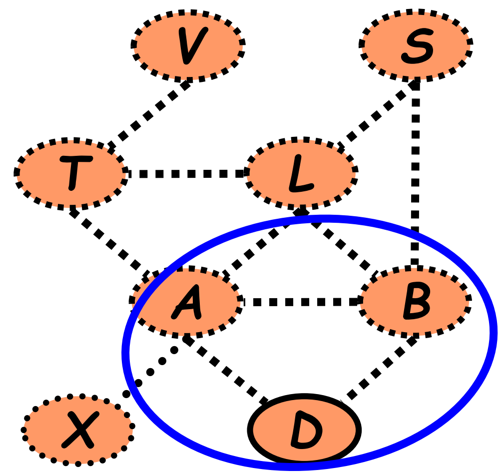
$$\tau_5(A,B) = \sum_L \tau_4(A,L)\tau_2(B,L)$$



Induced Graph in VE



- Want to compute $P(D)$
- Moralizing
- Eliminating V
- Eliminating S
- Eliminating X
- Eliminating T
- Eliminating L
- Eliminating A, B

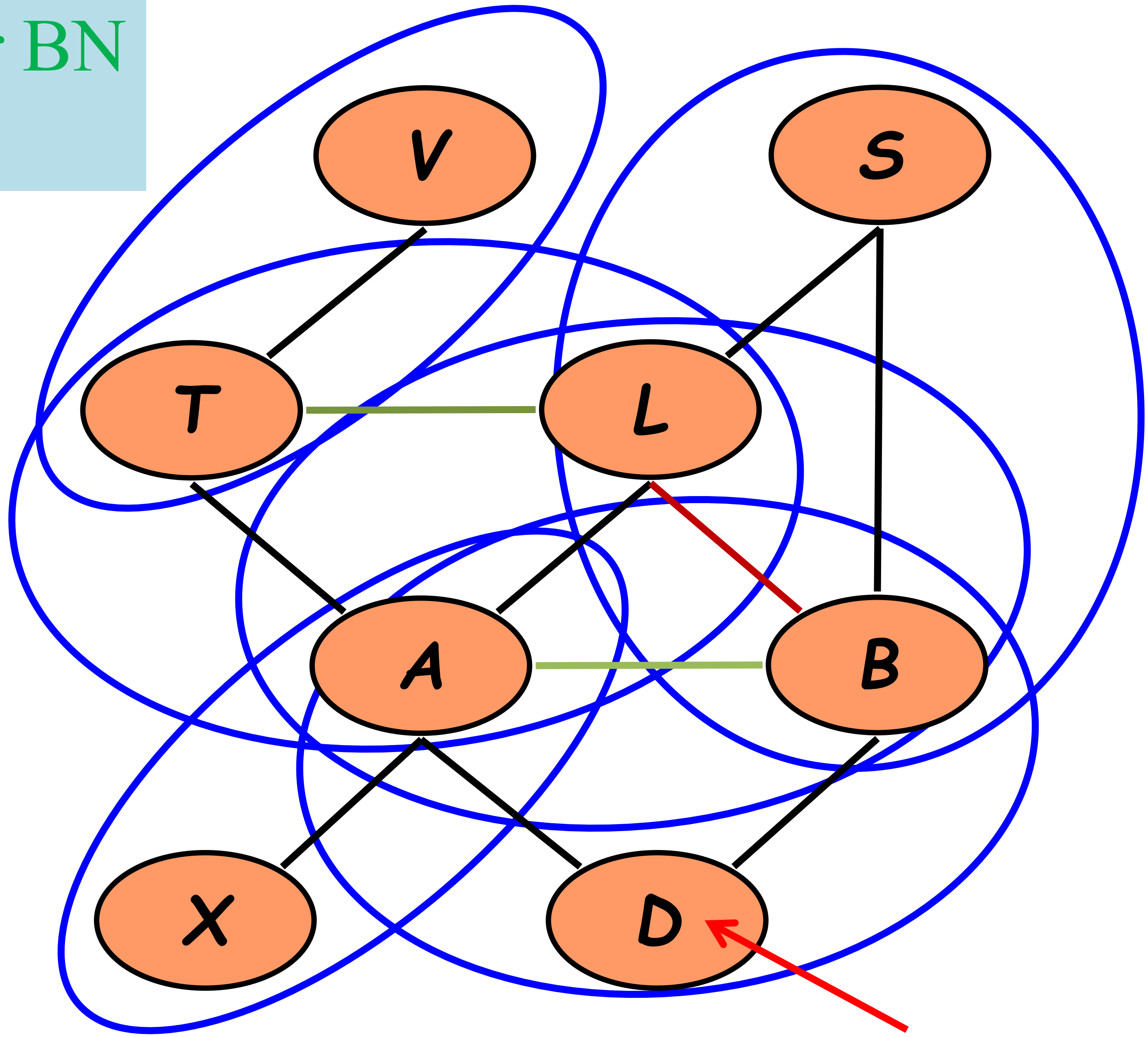


$$\tau_6(B,D) = \sum_A \tau_5(A,B)\tau_3(A)\phi_D(D,A,B)$$

$$\tau_7(D) = \sum_B \tau_6(B,D)$$

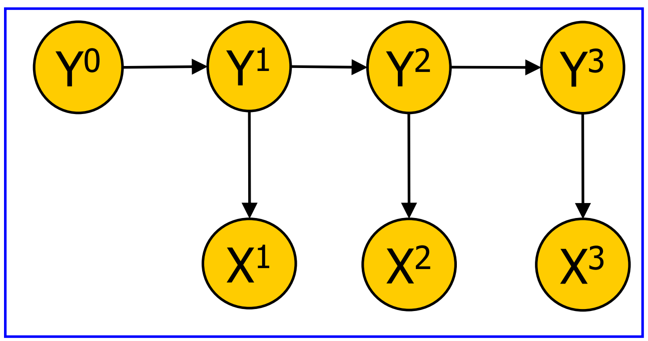
Induced Graph in VE

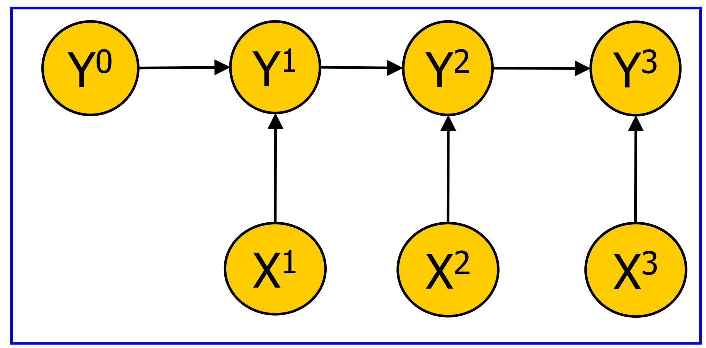
1) *Moralized* for BN
2) *Chordal*

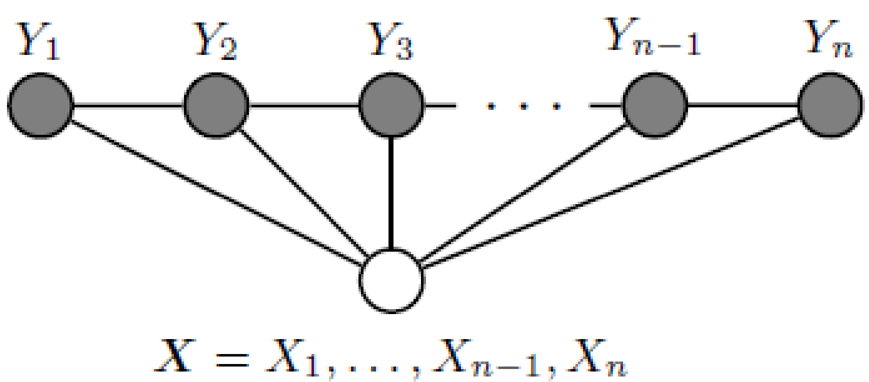


VE: Inferences in HMMs and CRFs

- Please recall the graphic representations of HMMs, MEMMs and linear-chain CRFs
- Given ***X***, the backbone of ***Y*** is the same



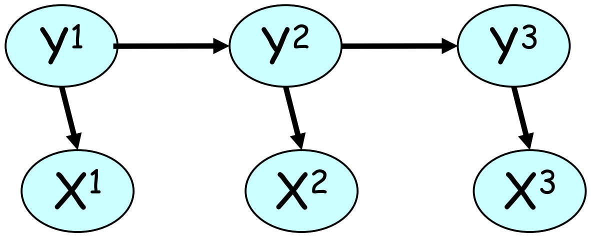




Compute $P(X|\theta)$

$$\alpha_t(i) = P(x_1, \ldots, x_t, y_t = i \mid \theta)$$

VE: Forward Algorithm



- Initialization:

$$\alpha_1(i) = \pi_i e_{i,x_1}$$

- Induction:

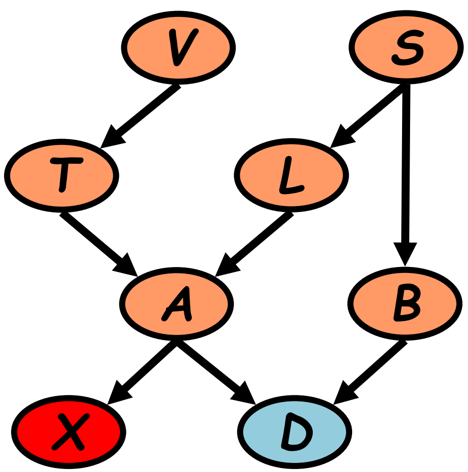
$$\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) t_{j,i}\right] e_{i,x_{t+1}}$$

- Termination:

$$P(X \mid \theta) = \sum_{i=1}^{N} \alpha_T(i)$$

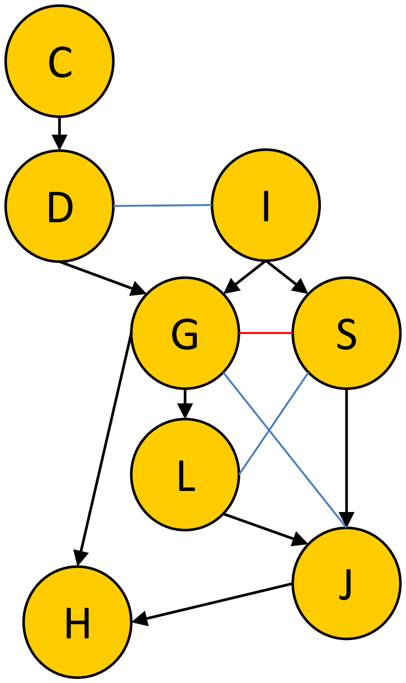
VE Disadvantages

- We need to traverse the whole graph for each run of reference
- Many intermediate results during previous runs of variable elimination can be re-used
  - For example, if we have run the reference of P(D), to infer P(X), results for eliminating V, S, T, L, A, B can be re-used
  - We can directly re-start from P(A)



VE Disadvantages

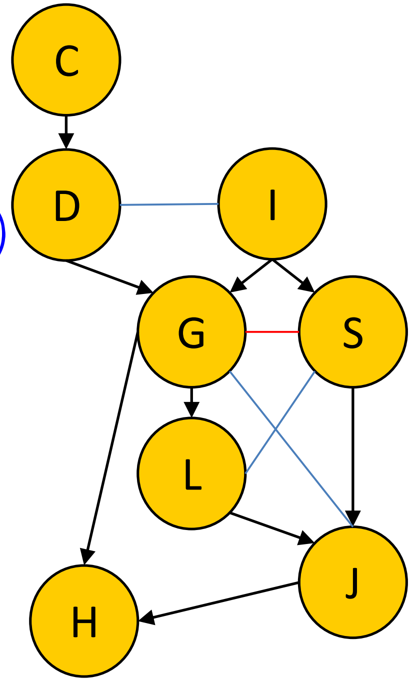
- For the induced undirected graph of BN (moralized & chordal), the basic structures are the cliques (maximal)
- If we can pre-calculate the marginal distributions defined on maximal cliques, the inferences may save many re-calculations



Can we design an algorithm to achieve this goal?

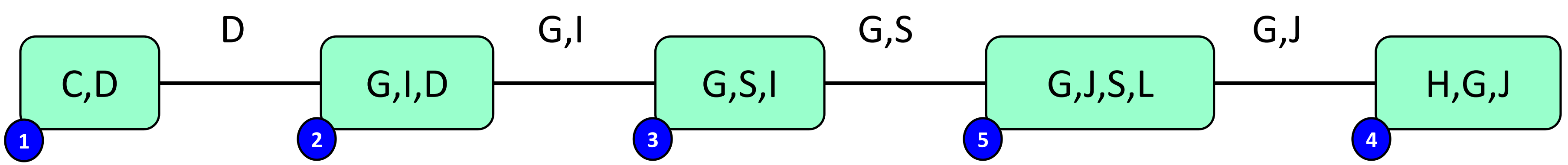
Clique Tree: A Concrete Example

- Clique Tree
  - For a chordal graph
  - A tree-like structure by cliques (maximal)



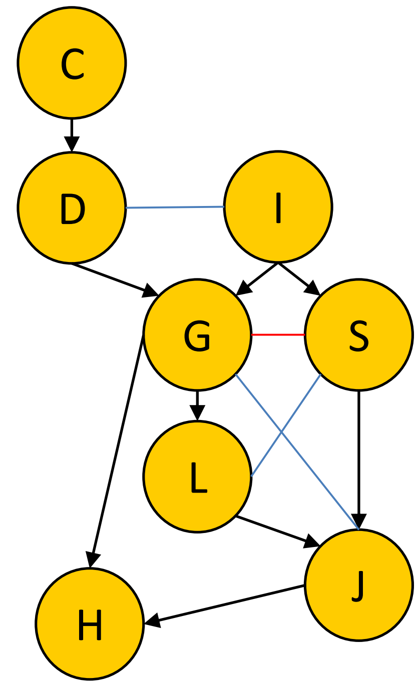
Two important features:

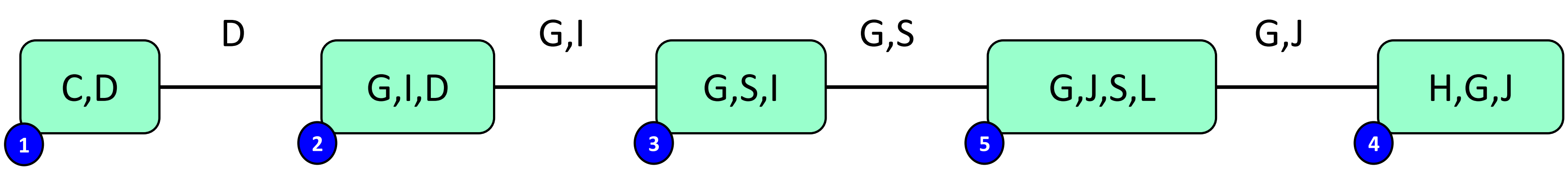
- Tree and family preserving
- Running intersection property



Clique Tree: A Concrete Example

- Tree and family preserving
  - **Factors** ϕ_i are defined on cliques
  - **Edges** are defined on the sepset $S_{i,j}$ of two directly connected cliques
- Running intersection property
  - Any variable X only exists in a **unique** sub-path along the tree

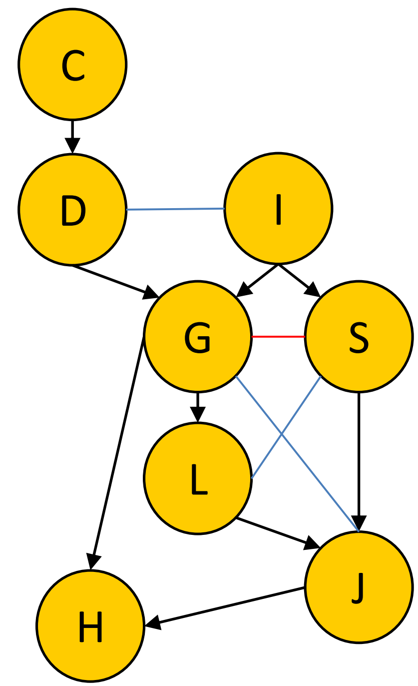


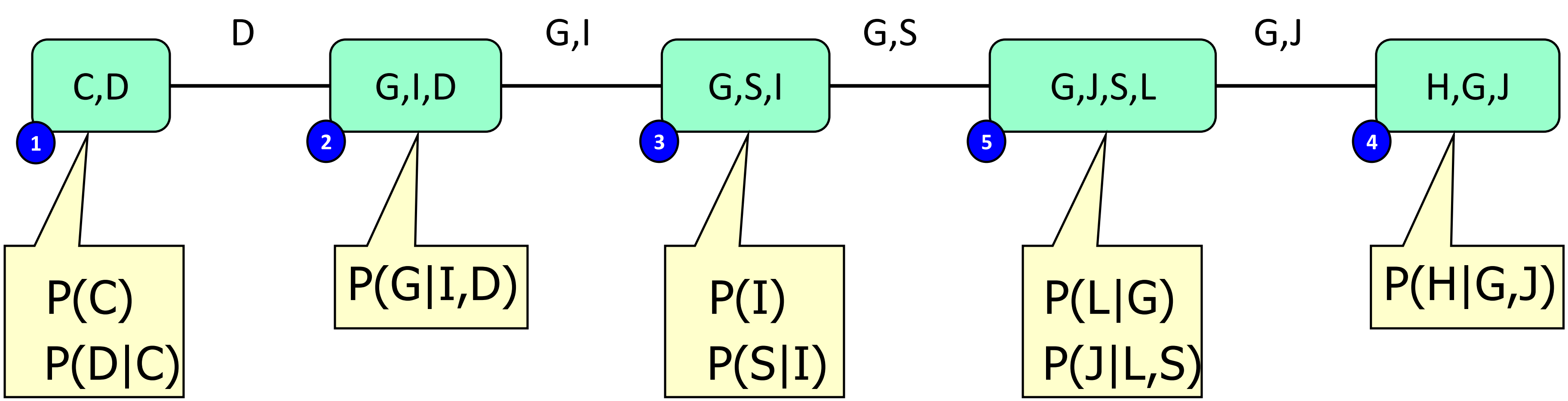


Clique Tree: A Concrete Example

- Assign local CPDs to **factors**

The **clique tree** is an equivalent representation of P as the original factorization representation



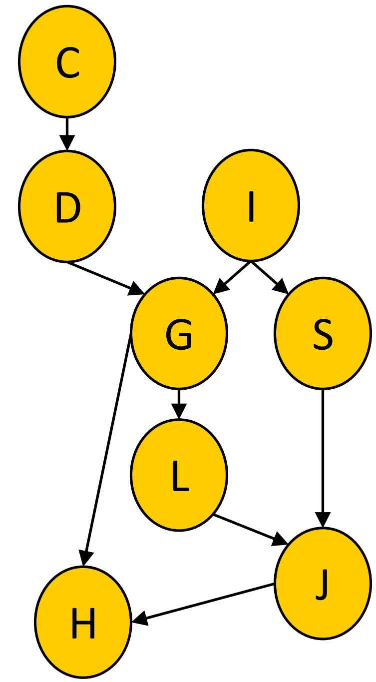


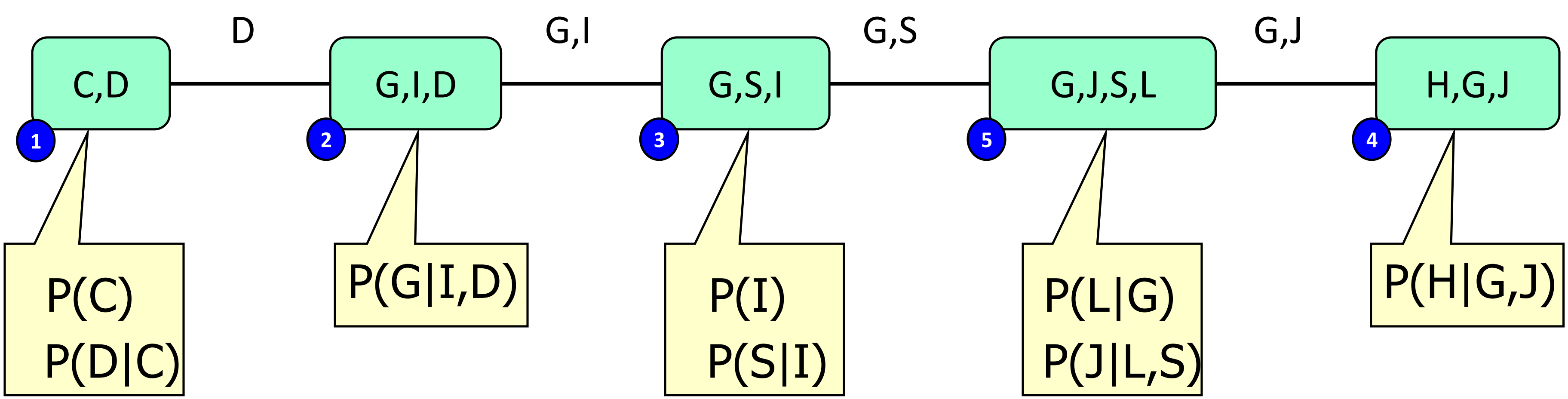
Exact Inference: Clique Trees

- Exploits factorization of the distribution for efficient inference, similar to variable elimination
  - Advantage: avoid unnecessary (or repeated) computations if repeated queries are needed
- Distribution (un-normalized) can be represented by **clique tree** with associated factors
  - $\tilde{P}_{\Phi}(\mathcal{X}) = \prod_{\phi_i \in \Phi} \phi_i(\mathcal{X}_i)$
    - For Bayesian networks, factors are local CPDs
    - For Markov networks, factors are clique potentials

Message Passing: Sum Product on Clique Tree

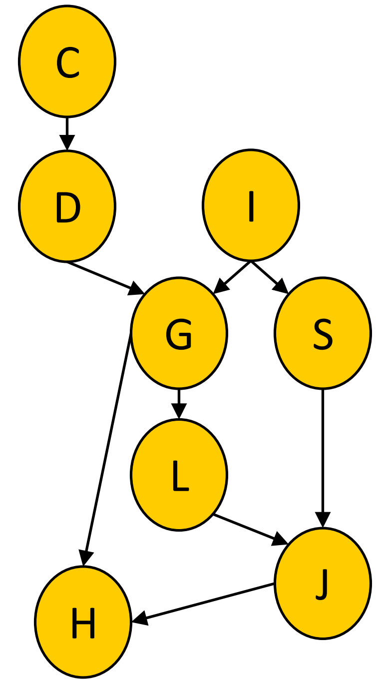
- Goal: Compute P(J)
  - Set initial factors at each cluster as products
  - C_1: Eliminate C, sending $\delta_{1\rightarrow 2}(D)$ to C_2
  - C_2: Eliminate D, sending $\delta_{2\rightarrow 3}(G,I)$ to C_3
  - C_3: Eliminate I, sending $\delta_{3\rightarrow 5}(G,S)$ to C_5
  - C_4: Eliminate H, sending $\delta_{4\rightarrow 5}(G,J)$ to C_5
  - C_5: Obtain P(J) by summing out G,S,L

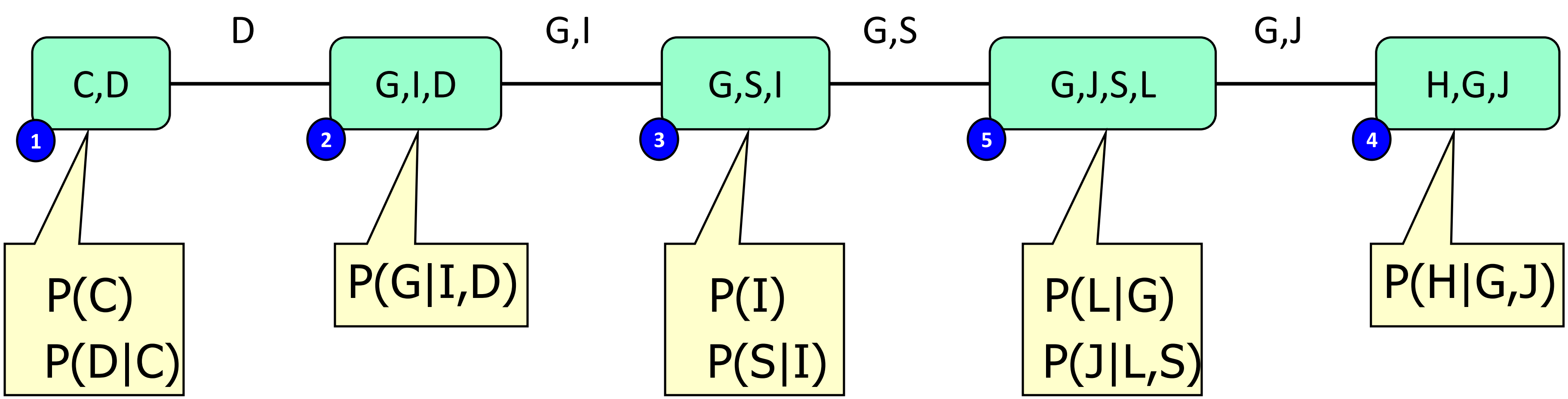


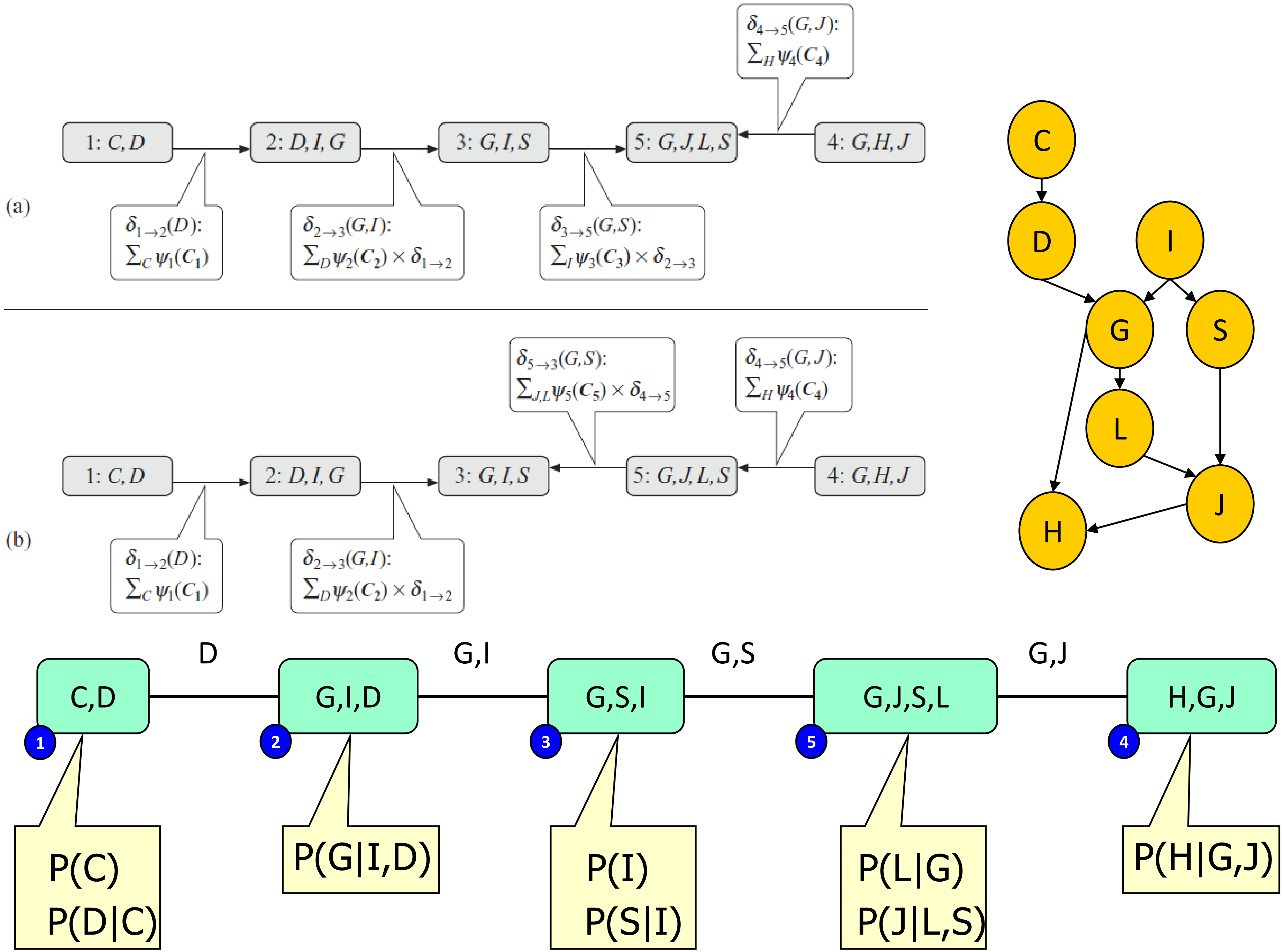


Message Passing: Sum Product on Clique Tree

- Goal: Compute P(J)
  - Set initial factors at each cluster as products
  - C_1: Eliminate C, sending $\delta_{1\to2}(D)$ to C_2
  - C_2: Eliminate D, sending $\delta_{2\to3}(G,I)$ to C_3
  - C_3: Eliminate I, sending $\delta_{3\to5}(G,S)$ to C_5
  - C_5: Eliminate SL, sending $\delta_{5\to4}(G,J)$ to C_4
  - C_4: Obtain P(J) by summing out H,G





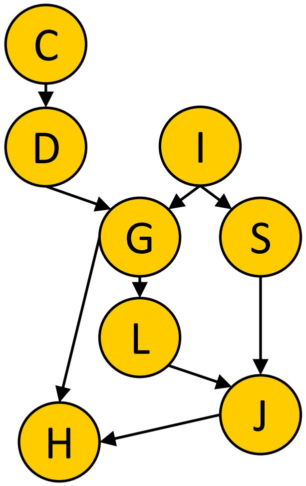


$\delta_{4\to5}(G,J)$: $\sum_H \psi_4(C_4)$
1: C, D
2: D, I, G
3: G, I, S
5: G, J, L, S
4: G, H, J
(a)
$\delta_{1\to2}(D)$: $\sum_C \psi_1(C_1)$
$\delta_{2\to3}(G,I)$: $\sum_D \psi_2(C_2) \times \delta_{1\to2}$
$\delta_{3\to5}(G,S)$: $\sum_I \psi_3(C_3) \times \delta_{2\to3}$
$\delta_{5\to3}(G,S)$: $\sum_{J,L} \psi_5(C_5) \times \delta_{4\to5}$
$\delta_{4\to5}(G,J)$: $\sum_H \psi_4(C_4)$
1: C, D
2: D, I, G
3: G, I, S
5: G, J, L, S
4: G, H, J
(b)
$\delta_{1\to2}(D)$: $\sum_C \psi_1(C_1)$
$\delta_{2\to3}(G,I)$: $\sum_D \psi_2(C_2) \times \delta_{1\to2}$
C
D
I
G
S
L
J
H
D
G,I
G,S
G,J
C,D
G,I,D
G,S,I
G,J,S,L
H,G,J
1
2
3
5
4
P(C)
P(D|C)
P(G|I,D)
P(I)
P(S|I)
P(L|G)
P(J|L,S)
P(H|G,J)

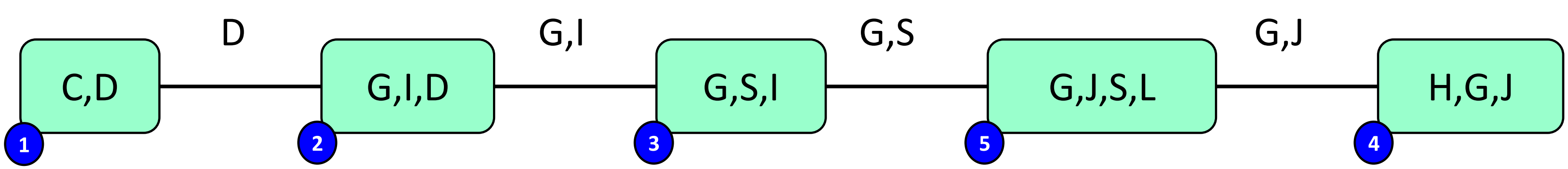
Clique Tree Message Passing

- Let T be a clique tree and $C_1,\ldots C_k$ its cliques
  - Multiply factors of each clique, resulting in initial potentials as each factor is assigned to some clique $\alpha(\phi)$ then we have

$$\beta_j^0[C_j] = \prod_{\phi:\alpha(\phi)=j} \phi \qquad \prod_{\phi} \phi = \prod_{j=1}^{k} \beta_j^0[C_j]$$

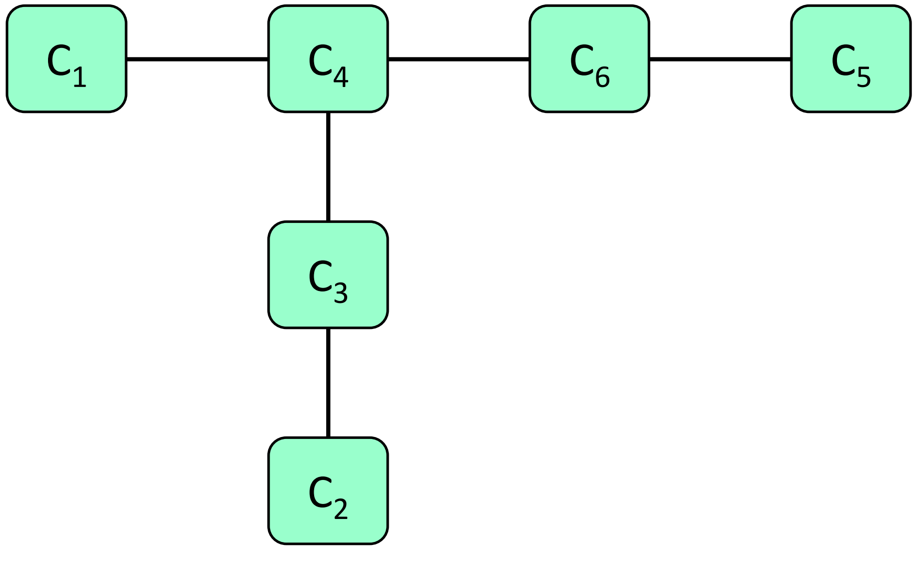


  - Define C_r as the root cluster
  - Start from tree leaves and move inward



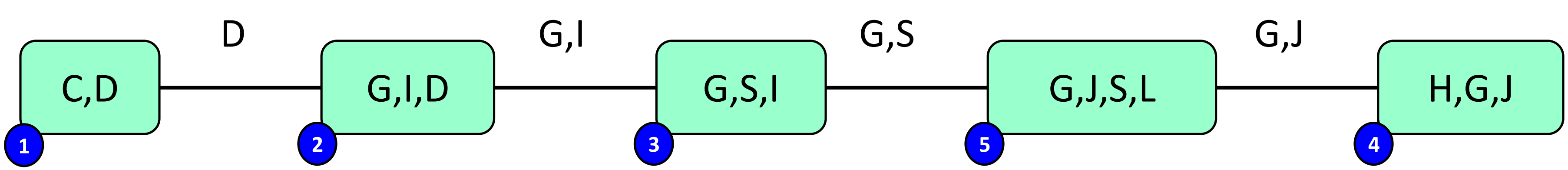
Clique Tree Message Passing: Example

- Root C_6
  - Legal ordering I: 1,2,3,4,5,6
  - Legal ordering II: 2,5,1,3,4,6
  - Illegal ordering: 3,4,1,2,5,6



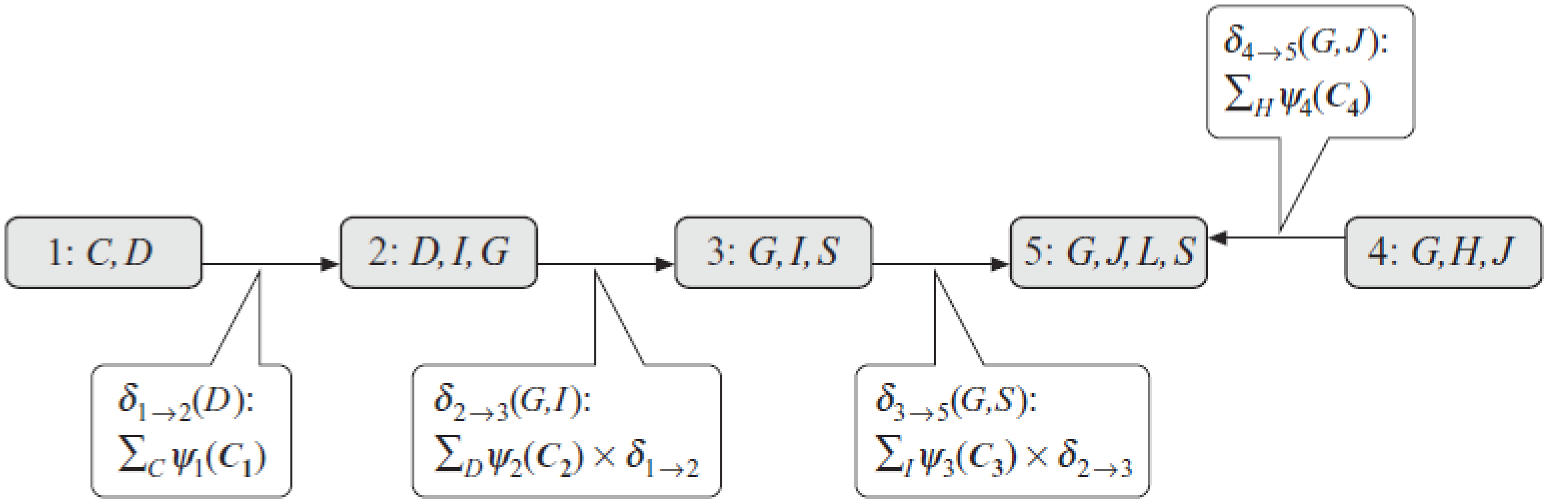
Clique Tree Calibration

- **Calibration** (校准): the sending messages of two adjacent cliques should be equal
- For calculating the probability of any variable, we need a more efficient algorithm to do the calculations rather than repeating above sum operations for each clique
- Obviously, there are some information during message passing which can be re-used to calculate the probability of other variables



Clique Tree Calibration

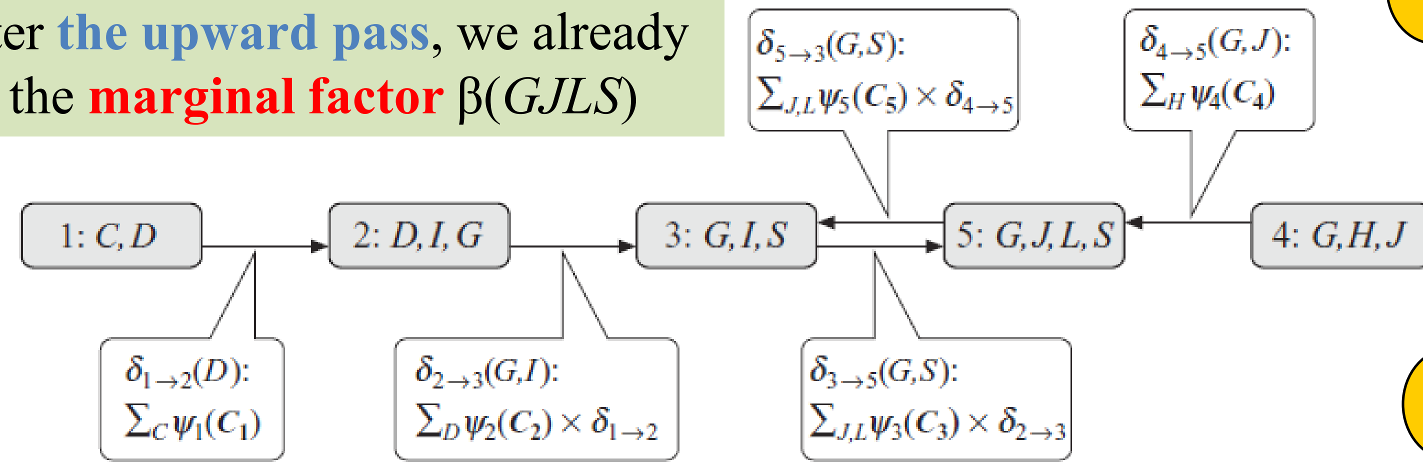
- We say that cluster C_i is ***ready*** to transmit to neighbor C_j when C_i has messages from all its neighbors except C_j.
- When C_i is ready, it can compute the message $\delta_{i\to j}(S_{i,j})$ by multiplying **its initial potential** β_i^0 (final potential β_i) with **all the coming messages** $\delta_{k\in\{Nb_i-j\}\to i}$ and then **eliminate the variables not in the sepset** $C_i - S_{i,j}$

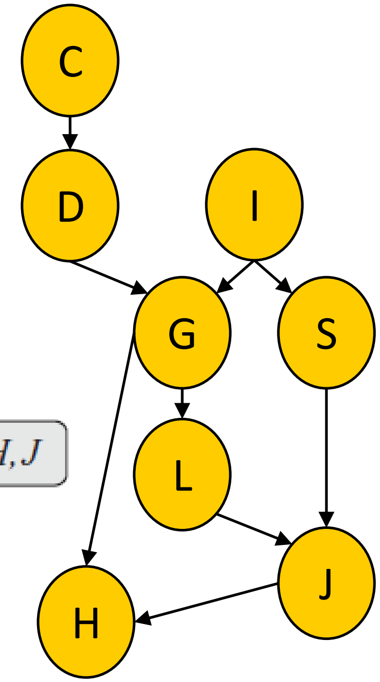


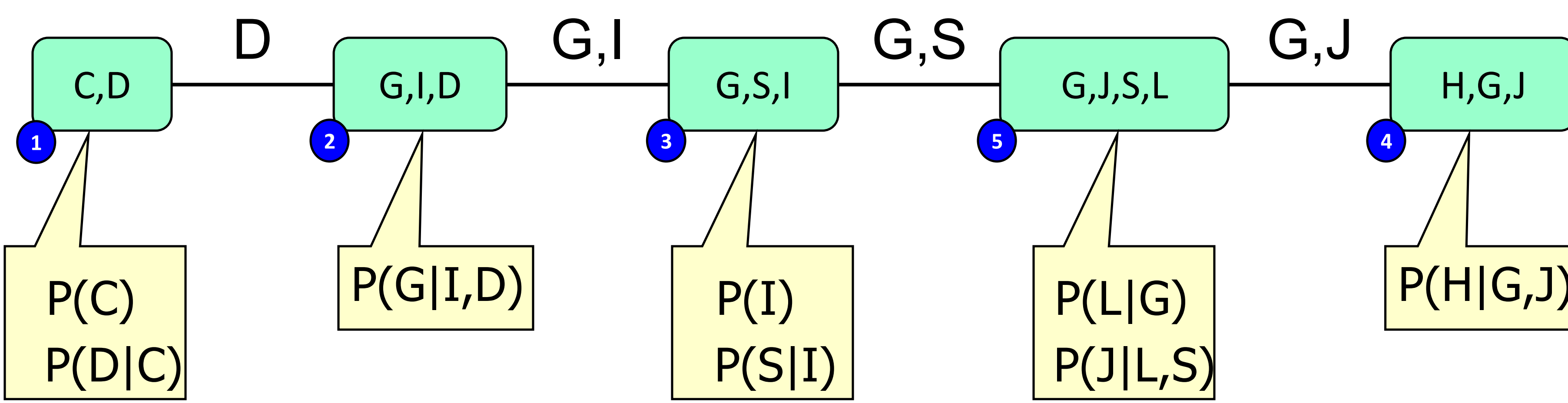
Clique Tree Calibration: Example

- Root: C_5 (first downward pass)

After **the upward pass**, we already get the **marginal factor** $\beta(GJLS)$

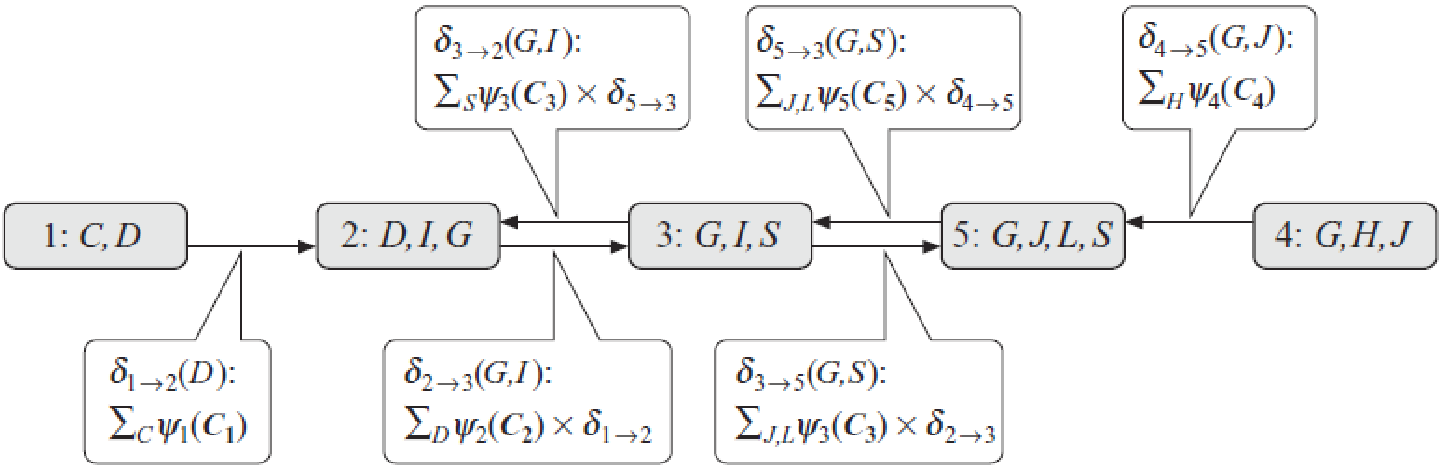


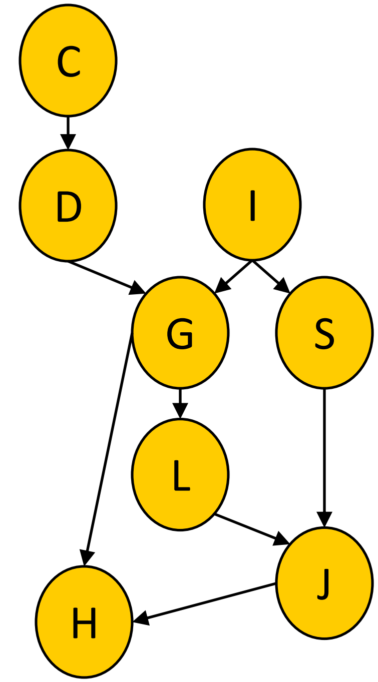


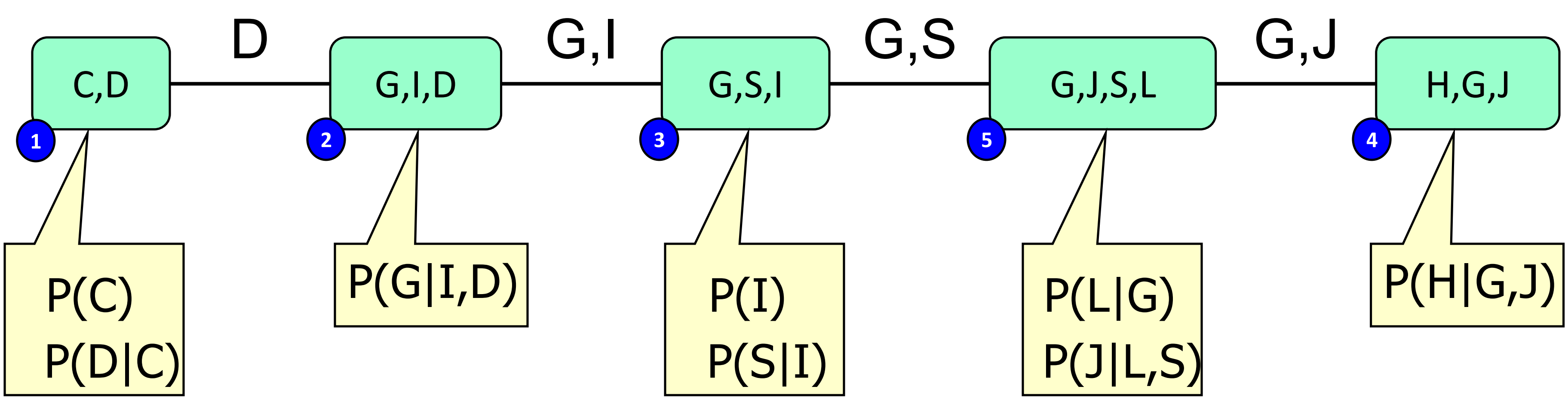


Clique Tree Calibration: Example

- Root: C_5 (second downward pass)







Clique Tree Calibration: Sum-Product

Algorithm 10.2 Calibration using sum-product message p

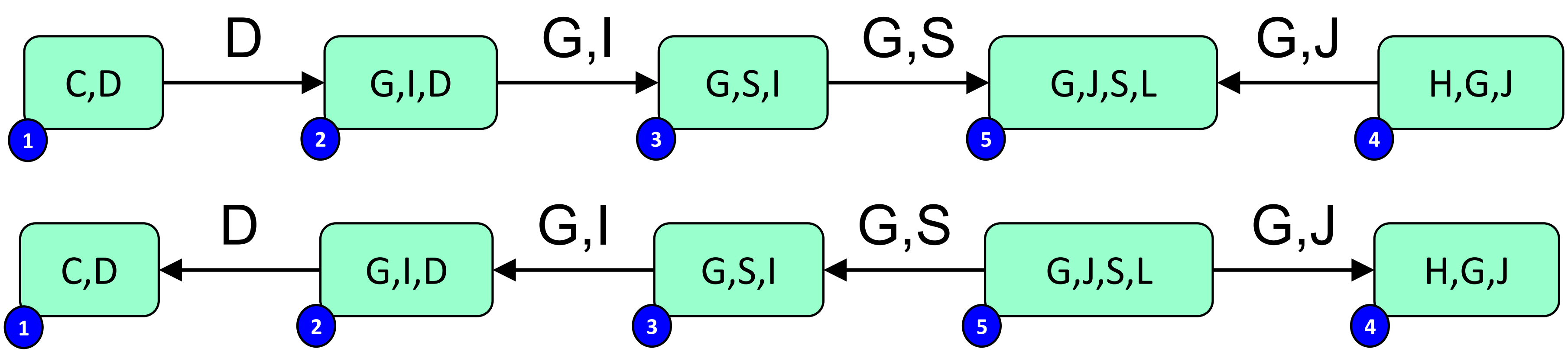
Procedure CTree-SP-Calibrate (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)

Initialize-Cliques
while exist i, j such that i is ready to transmit to j
  $\delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow$ SP-Message(i, j)
for each clique i
  $\beta_i \leftarrow \psi_i \cdot \prod_{k \in \mathrm{Nb}_i} \delta_{k \to i}$
return $\{\beta_i\}$

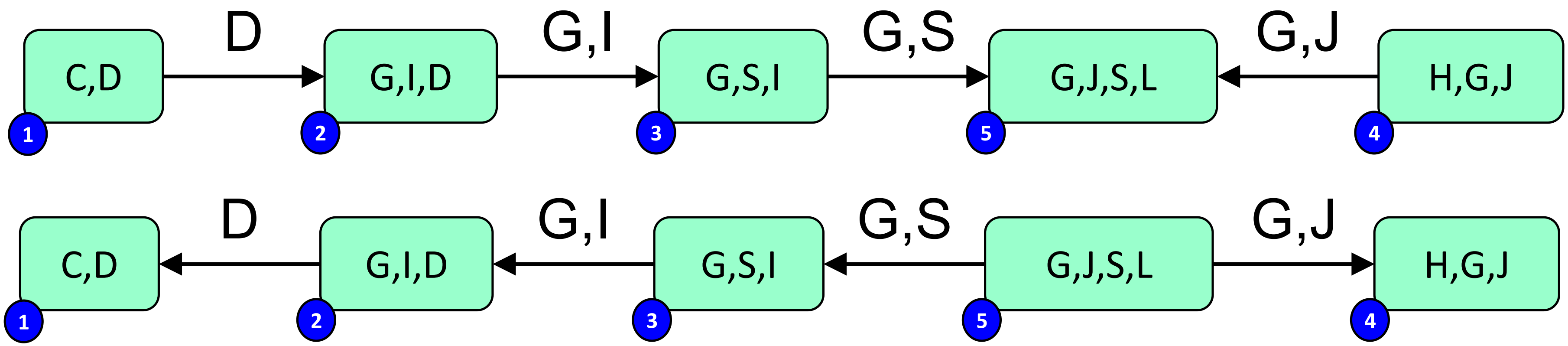
Procedure Initialize-Cliques (

)
  for each clique $\boldsymbol{C}_i$
    $\psi_i(\boldsymbol{C}_i) \leftarrow \prod_{\phi_j \,:\, \alpha(\phi_j)=i} \phi$

Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)
  $\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i}$
  $\tau(\boldsymbol{S}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi(\boldsymbol{C}_i)$
  return $\tau(\boldsymbol{S}_{i,j})$



Clique Tree Calibration: Sum-Product



- After **calibration**, the **final factors associated with cliques** are updated to **the marginal distributions (or factors)** over the cliques

Clique Tree Calibration

- If X appears in multiple cliques, they must agree
  - A clique tree with potentials $\beta_i[C_i]$ is said to be **calibrated** if for all neighboring cliques C_i and C_j:
  - $\sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$

- Advantage: compute posteriors for all the cliques using only twice passes

Calibrated Clique Tree as Distribution

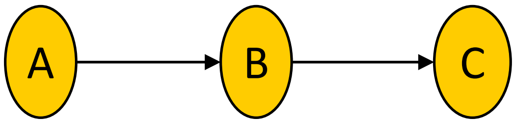
- A calibrated clique tree is more than simply a data structure that stores the results of probabilistic inference for all of the clique in the tree.
- It can also be viewed as an alternative representation of the measure $\tilde{P}_{\Phi}$ (un-normalized distribution).
- We can easily prove:

The product of the marginal distributions (or factors) of all the cliques divided by the marginal factors of all the sepsets

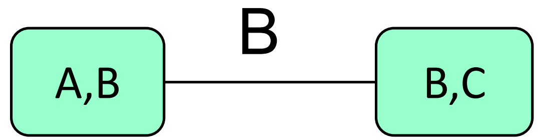
$$\frac{\prod_{i} \beta_i(C_i)}{\prod_{(C_i \leftrightarrow C_j)\in T} \mu_{i,j}(S_{i,j})}$$

Calibrated Clique Tree as Distribution

Bayesian network



Clique tree



- For calibrated tree

$$P(C \mid B) = \frac{P(B,C)}{P(B)} = \frac{\beta_2[B,C]}{P(B)} = \frac{\beta_2[B,C]}{\sum_C \beta_2[B,C]} = \frac{\beta_2[B,C]}{\sum_A \beta_1[A,B]}$$

- Joint distribution can thus be written as

$$P(A,B,C) = P(A,B)P(C \mid B) = \frac{\beta_1[A,B]\beta_2[B,C]}{\mu_{1,2}[B]}$$

Calibrated trees can be alternative representations of the distributions.

They are equal!

To Random-Order Message Passing

- In the above sum-product message passing, a factor i can transmit a message to j only if it has received all the other messages (factor is ready)

- Can the factor i transmit a message to its neighbors when it is not ready?

Message Passing: Belief Update

- Recall the clique tree calibration algorithm
  - Upon calibration the final potential at *i* is:

$$\beta_i = \beta_i^0 \prod_{k \in N_i} \delta_{k \to i}$$

  - A message from *i* to *j* sums out the non-sepset variables from the product of initial potential and all other messages

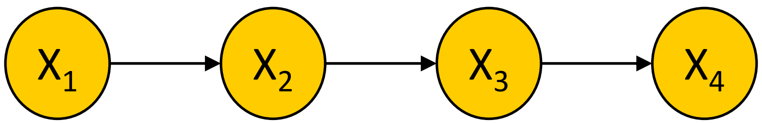
$$\delta_{i \to j} = \sum_{C_i - S_{i,j}} \beta_i^0 \prod_{k \in N_i - \{j\}} \delta_{k \to i}$$

  - Can also be viewed as multiplying all messages and dividing by the message from j to i

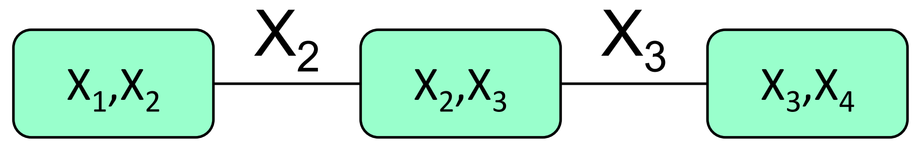
$$\delta_{i \to j} = \frac{\sum_{C_i - S_{i,j}} \beta_i^0 \prod_{k \in N_i} \delta_{k \to i}}{\delta_{j \to i}} = \frac{\sum_{C_i - S_{i,j}} \pi_i}{\delta_{j \to i}}$$

Message Passing: Belief Update

Bayesian network



Clique tree



- Root: C_2
- C_1 to C_2 Message: $\delta_{1\to 2}(X_2) = \sum_{X_1} \beta_1^0[X_1, X_2] = \sum_{X_1} P(X_1)P(X_2 \mid X_1)$
- C_2 to C_1 Message: $\delta_{2\to 1}(X_2) = \sum_{X_3} \beta_2^0[X_2, X_3]\delta_{3\to 2}(X_3)$
- Alternatively compute $\beta_2[X_2, X_3] = \delta_{1\to 2}(X_2)\delta_{3\to 2}(X_3)\beta_2^0[X_2, X_3]$
- And then:

$$\delta_{2\to 1}(X_2) = \frac{\sum_{X_3} \beta_2[X_2, X_3]}{\delta_{1\to 2}(X_2)} = \sum_{X_3} \frac{\beta_2[X_2, X_3]}{\delta_{1\to 2}(X_2)} = \sum_{X_3} \beta_2^0[X_2, X_3]\delta_{3\to 2}(X_3)$$

- → Thus, the two approaches are equivalent

Message Passing: Belief Update

- Based on the observation above, belief update
  - Different message passing scheme
  - Each clique C_i maintains its fully updated beliefs β_i
    - product of initial messages and messages from neighbors
  - Store at each sepset $S_{i,j}$ the previous message $\mu_{i,j}$ passed regardless of the direction
  - When passing a message, divide by previous $\mu_{i,j}$

  - Claim: message passing is correct regardless of the clique that sent the last message
  - This is called belief update

Algorithm for Belief Update

Algorithm 10.3 Calibration using belief propagation in clique tree

Procedure CTree-BU-Calibrate (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)
  Initialize-CTree
  while exists an uninformed clique in $\mathcal{T}$
    Select $(i\text{–}j) \in \mathcal{E}_{\mathcal{T}}$
    BU-Message(i, j)
  return $\{\beta_i\}$

Procedure Initialize-CTree (
)
  for each clique $\boldsymbol{C}_i$
    $\beta_i \leftarrow \prod_{\phi \,:\, \alpha(\phi)=i} \phi$
  for each edge $(i\text{–}j) \in \mathcal{E}_{\mathcal{T}}$
    $\mu_{i,j} \leftarrow 1$

The BU is arbitrary. You can randomly choose any edge in the clique graph for the update. At convergence:

$$\sum_{C_i - S_{i,j}} \beta_i = \sum_{C_j - S_{i,j}} \beta_j$$

Procedure BU-Message (
  i, // sending clique
  j // receiving clique
)
  $\sigma_{i \to j} \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i$
    // marginalize the clique over the sepset
  $\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i \to j}}{\mu_{i,j}}$
  $\mu_{i,j} \leftarrow \sigma_{i \to j}$

Clique Tree Invariant for BU

- BU maintains **distribution invariant property**
  - Upon calibration we have $$\boxed{P(\mathrm{U}) = \frac{\prod_{C_i \in T} \beta_i[C_i]}{\prod_{(C_i \leftrightarrow C_j) \in T} \mu_{i,j}(S_{i,j})}}$$
  - Initially this invariant holds obviously
  - At each update step invariant is also maintained
    - Message only changes π_j and $\mu_{i,j}$
    - We need to prove $\frac{\beta_j^U}{\mu_{i,j}^U} = \frac{\beta_j}{\mu_{i,j}}$
    - This is exactly the message passing step $\beta_j^U = \beta_j \mu_{i,j}^U / \mu_{i,j}$

Belief update *re-parameterizes* P at each step

Inference on Calibrated Clique Trees

- Single variable inference
  - The posterior of a target variable X can be directly computed by eliminating the redundant variables from a clique that contains X
- Inference outside a clique
- Inference with increment (or evidence)

After calibration, **the final factors associated with cliques** are updated to **the marginal distributions** (or factors) over the cliques

Answering Queries Outside a Clique

Algorithm 10.4 Out-of-clique inference in clique tree

Procedure CTree-Query (
  $\mathcal{T}$, // Clique tree over Φ
  $\{\beta_i\}, \{\mu_{i,j}\}$, // Calibrated clique and sepset beliefs for $\mathcal{T}$
  $\boldsymbol{Y}$ // A query
)

Let $\mathcal{T}'$ be a subtree of $\mathcal{T}$ such that $\boldsymbol{Y} \subseteq Scope[\mathcal{T}']$
Select a clique $r \in \mathcal{V}_{\mathcal{T}'}$ to be the root
$\Phi \leftarrow \beta_r$
for each $i \in \mathcal{V}'_{\mathcal{T}}$
  $\phi \leftarrow \frac{\beta_i}{\mu_{i,p_r(i)}}$
  $\Phi \leftarrow \Phi \cup \{\phi\}$
$\boldsymbol{Z} \leftarrow Scope[\mathcal{T}'] - \boldsymbol{Y}$
Let $\prec$ be some ordering over $\boldsymbol{Z}$
return Sum-Product-VE($\Phi, \boldsymbol{Z}, \prec$)

$$P(Y') = \frac{\prod_{i \in V_{T'}} \beta_i}{\prod_{(i-j) \in \varepsilon_{T'}} \mu_{i,j}}$$

$$Y \subseteq Y' = Scope(T')$$

Find **a minimal sub-path** on the clique tree which contains all the query variables!

Answering Queries with Increments

- Introducing evidence $Z=z$

- Compute posterior of X where X is in a clique with Z
  - Since clique tree is calibrated, multiply the clique that contains X and Z with indicator function $I(Z=z)$ and sum out irrelevant variables

- Compute posterior of X if not sharing a clique with Z
  - Introduce indicator function $I(Z=z)$ into some clique containing Z and propagate messages along path to clique containing X
  - Sum out irrelevant factors from clique containing X

Comments on Calibrated Clique Trees

- What is a calibrated clique tree?
  - From the initial factors $\psi(C_i)$ or ψ_i generated from BNs/MNs, a clique tree is calibrated if we get the joint distributions $\beta(C_i)$ or β_i associated with all nodes (cliques) and the joint distributions $\mu_{i,j}$ of all sepsets in the tree. We can use either **sum product** or **belief update** to do the calibration.

Comments on Calibrated Clique Trees

- What is a calibrated clique tree?

- Why does a clique tree need to be calibrated?
  - All CPDs/factors in BNs/MNs are equivalently transformed as calibrated factors and messages in clique tree. The joint distribution is invariant for the calibrated beliefs and all the steps of BU.
  - The calibrated factors β_i and messages $\mu_{i,j}$ are the marginal distributions of the variables defined on the factors/sepsets in the clique tree.

Comments on Calibrated Clique Trees

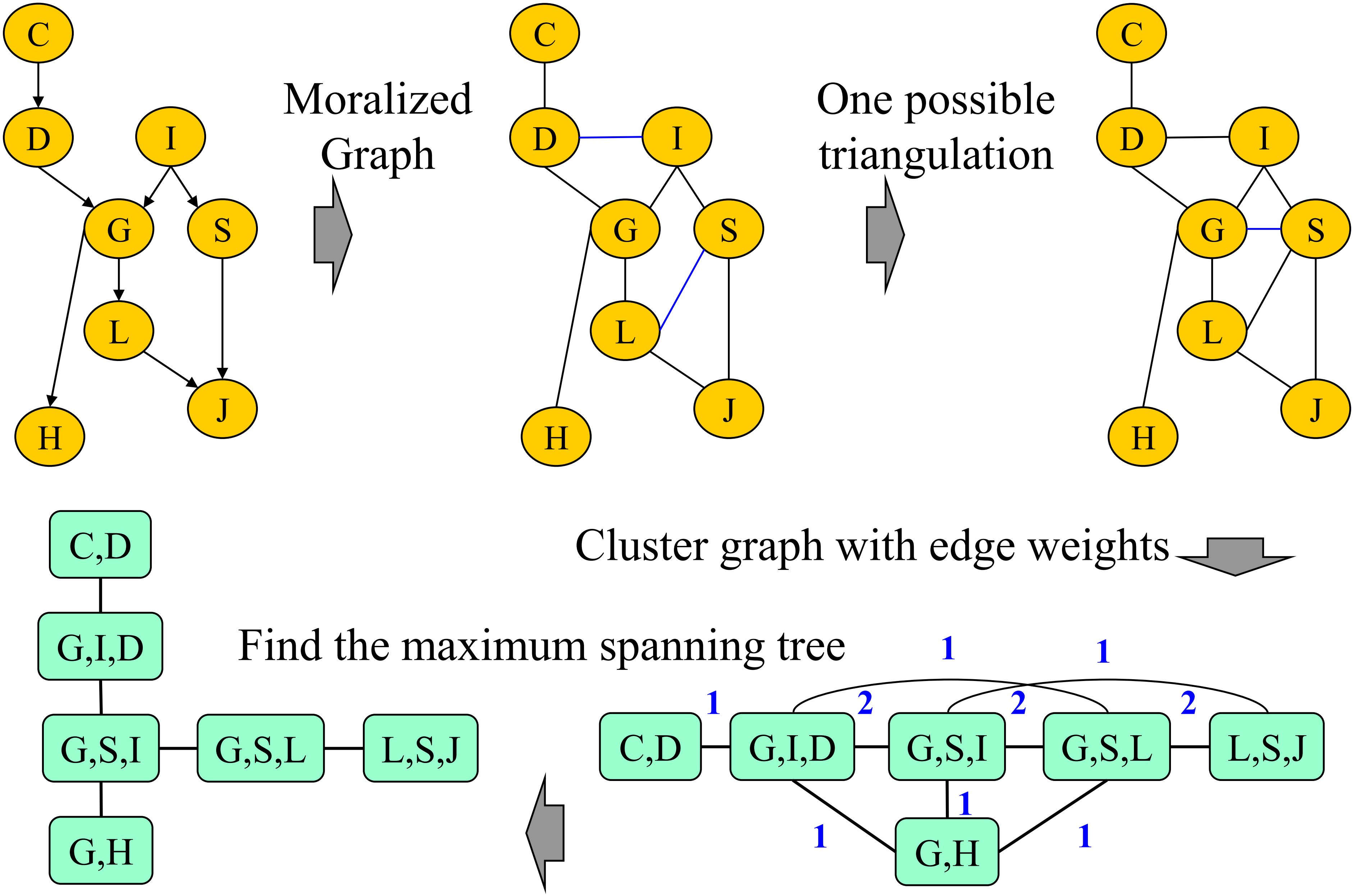
- What is a calibrated clique tree?
- Why does a clique tree need to be calibrated?

- What is the advantage of clique tree?
  - In most cases, the structure of clique tree is simpler than the original BNs/MNs. Inference will be more efficient.
  - *Belief propagation* can be easily extended to approximate inference

Constructing Clique Trees

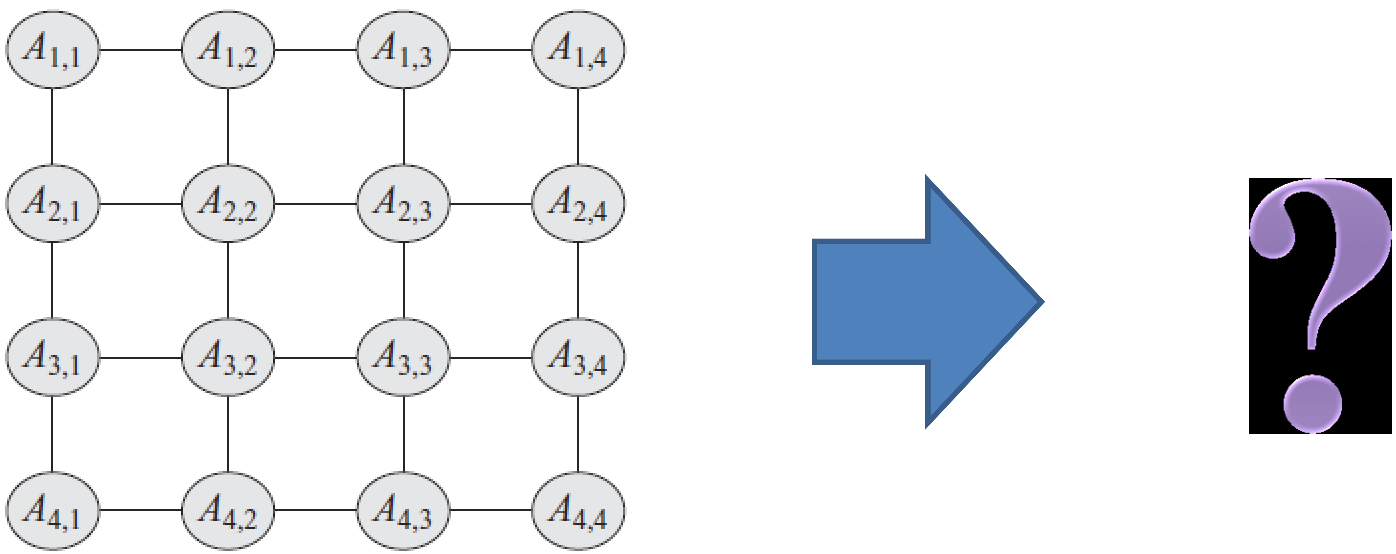
- Goal: construct a tree that is family preserving and obeys the running intersection property

- **Triangulate** the graph to construct a chordal graph H
  - NP-hard to find triangulation where the largest clique in the resulting chordal graph has minimum size
- Find cliques in H and make each a node in the graph
  - Finding maximal cliques is NP-hard
  - Can start with families and grow greedily
- Construct a tree over the clique nodes
  - Use maximum spanning tree on an undirected graph whose nodes are maximal cliques and edge weight is $|C_i \cap C_j|$
  - Can show that resulting graph obeys running intersection


C
D
I
G
S
L
H
J
Moralized
Graph
C
D
I
G
S
L
H
J
One possible
triangulation
C
D
I
G
S
L
H
J
Cluster graph with edge weights
C,D
G,I,D
G,S,I
G,S,L
L,S,J
G,H
Find the maximum spanning tree
1
1
1
2
2
2
C,D
G,I,D
G,S,I
G,S,L
L,S,J
1
1
1
G,H

Limitations of Clique Tree

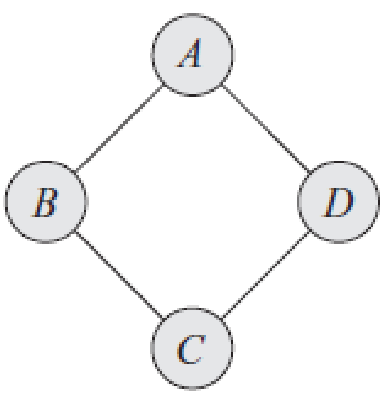
- Hard to construct induced graph & clique tree in large graph
- Inefficiency for Markov networks with loops



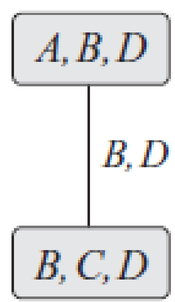
Pairwise Markov Network

From Clique Tree to Loopy Cluster Graph

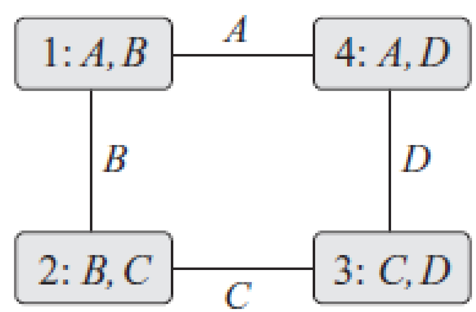
- Can we directly define clusters on the cliques of the original rather than induced graph?
- The message passing strategy cannot stop due to the problem of *message circulating*



Markov Network



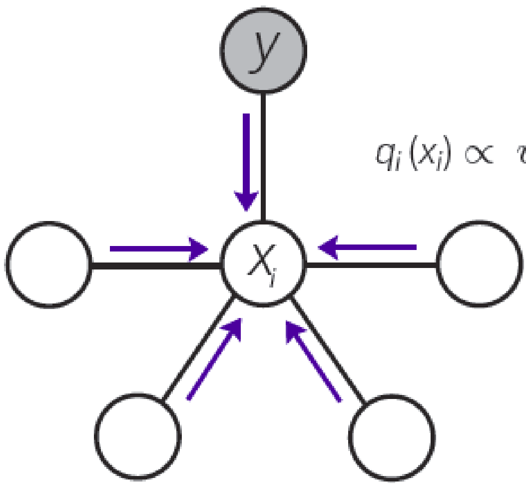
Clique Tree



Loopy Cluster Graph

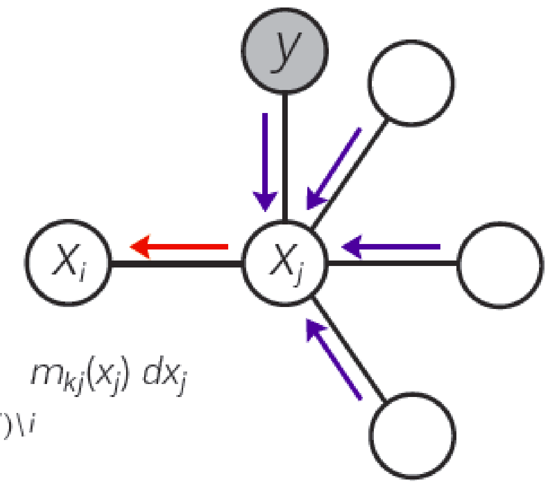
BP in pairwise MRF using Loopy CG

Figure 2. Message-passing recursions underlying the BP algorithm. ***Left:*** **Approximate marginal (belief) estimates combine the local observation potential with messages from neighboring nodes.** ***Right:*** **A new outgoing message (red) is computed from all other incoming messages (blue).**



$$q_i(x_i) \propto \psi_i(x_i, y) \prod_{j\in\Gamma(i)} m_{ji}(x_i)$$

$$m_{ji}(x_i) \propto \int_{\mathcal{X}_j} \psi_{ij}(x_i, x_j)\psi_j(x_j, y) \prod_{k\in\Gamma(j)\setminus i} m_{kj}(x_j)\, dx_j$$

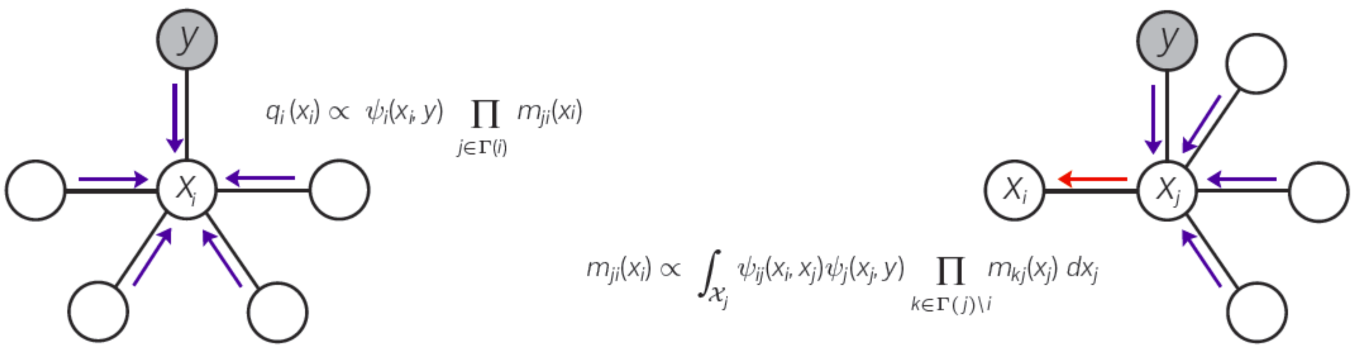


Approximate marginal by multiplying initial beliefs and all the incoming messages

A new message by multiplying initial beliefs and all the incoming messages except *i*

BP in pairwise MRF using Loopy CG

Figure 2. Message-passing recursions underlying the BP algorithm. ***Left:*** **Approximate marginal (belief) estimates combine the local observation potential with messages from neighboring nodes.** ***Right:*** **A new outgoing message (red) is computed from all other incoming messages (blue).**



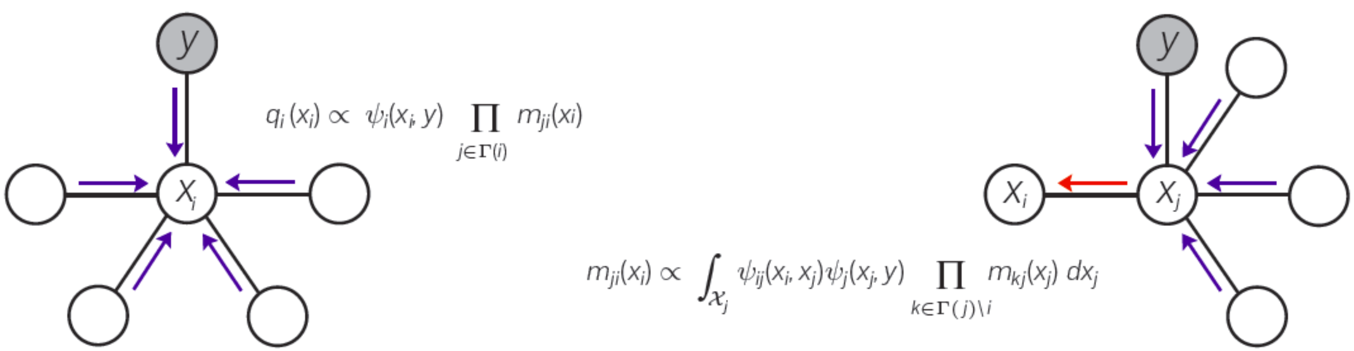
Input: node potentials $\psi_s(x_s)$, edge potentials $\psi_{st}(x_s, x_t)$;
Initialize messages $m_{s \to t}(x_t) = 1$ for all edges $s - t$;
Initialize beliefs $\mathrm{bel}_s(x_s) = 1$ for all nodes s;
repeat
  Send message on each edge
  $m_{s \to t}(x_t) = \sum_{x_s} \left(\psi_s(x_s)\psi_{st}(x_s, x_t) \prod_{u \in \mathrm{nbr}_s \setminus t} m_{u \to s}(x_s) \right)$;
  Update belief of each node $\mathrm{bel}_s(x_s) \propto \psi_s(x_s) \prod_{t \in \mathrm{nbr}_s} m_{t \to s}(x_s)$;
until *beliefs don't change significantly*;
Return marginal beliefs $\mathrm{bel}_s(x_s)$;

NOTE: ***normalize to 1*** after each iteration

BP in pairwise MRF using Loopy CG

- Let’s show an example

Figure 2. Message-passing recursions underlying the BP algorithm. ***Left:*** **Approximate marginal (belief) estimates combine the local observation potential with messages from neighboring nodes.** ***Right:*** **A new outgoing message (red) is computed from all other incoming messages (blue).**



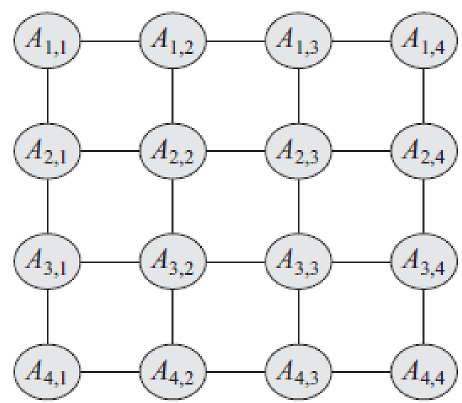
BP in Loopy Cluster Graph (General)

Procedure CGraph-SP-Calibrate (
  Φ, // Set of factors
  $\mathcal{U}$ // Generalized cluster graph Φ
)
  Initialize-CGraph
  $\boxed{\begin{array}{l} \textbf{while } \text{graph is not calibrated} \\ \quad \text{Select } (i\text{–}j) \in \mathcal{E}_{\mathcal{U}} \\ \quad \delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow \text{SP-Message}(i, j) \end{array}}$
  for each clique i
    $\beta_i \leftarrow \psi_i \cdot \prod_{k \in \mathrm{Nb}_i} \delta_{k \to i}$
  return $\{\beta_i\}$

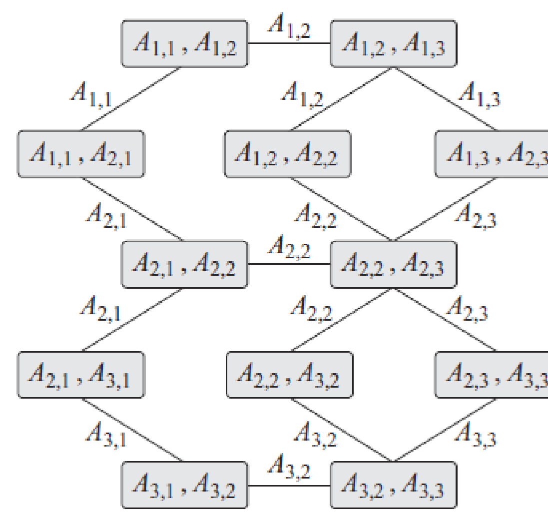
Procedure Initialize-CGraph (
  $\mathcal{U}$
)
  for each cluster $\boldsymbol{C}_i$
    $\beta_i \leftarrow \prod_{\phi \,:\, \alpha(\phi)=i} \phi$
  for each edge $(i\text{–}j) \in \mathcal{E}_{\mathcal{U}}$
    $\delta_{i \to j} \leftarrow 1$
    $\delta_{j \to i} \leftarrow 1$

Put ***initial factors*** into different ***cliques***

Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)
  $\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (\mathrm{Nb}_i - \{j\})} \delta_{k \to i}$
  $\tau(\boldsymbol{S}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi(\boldsymbol{C}_i)$
  return $\tau(\boldsymbol{S}_{i,j})$



Markov Network



Loopy Cluster Graph

Comments for BP in Loopy Graph

- Initialize all the beliefs and messages as 1, so any cluster can send its messages at the beginning
- Don't require messages are equal in both directions
- Maintain the factor beliefs by multiplying initial beliefs/potentials with all the incoming messages
- Send messages by multiplying initial beliefs/potentials with all the incoming messages except the target cluster (eliminating the beliefs of the variables not in the sepset)

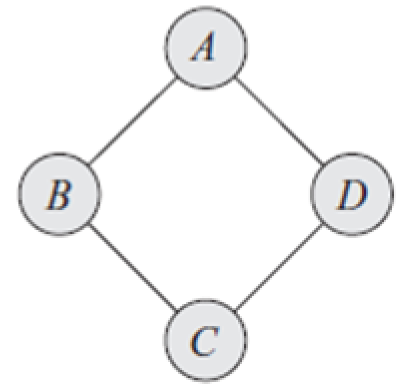
Please read the #2 Textbook, Chapter 22.2.2, Page 770

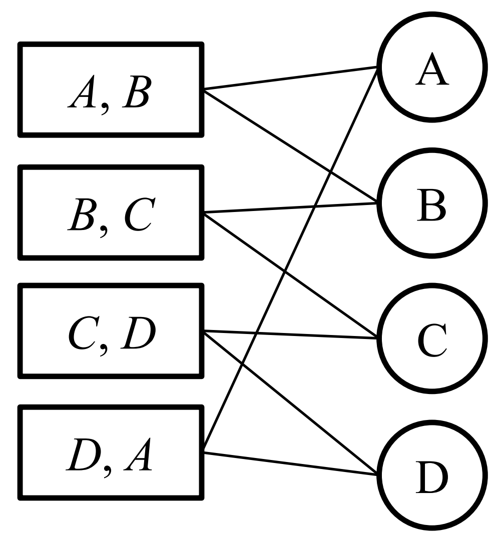
Running Intersection Property

- For any variable, if a variable X exists in cluster C_i and C_j, there exists only a single path between the two clusters, all the clusters on the path contain the variable X
- If a cluster graph follows running intersection property, the calibrated beliefs are *approximate marginal potentials of the clusters*
  - Beliefs do not change over time
  - Or $\sum_{C_i - S_{i,j}} \beta_i = \sum_{C_j - S_{i,j}} \beta_j$

Bethe Cluster Graph or Factor Graph

- Two types of nodes
  - Factor nodes defined on cliques
  - Variable nodes defined on variables
- Factor graph ensures running interaction property
- Two types of messages
  - From factors to variables (simply eliminate other variables)
  - From variables to factors





Bethe graph can easily calculate the *approximate marginal* of single variable

The Problem of Convergence

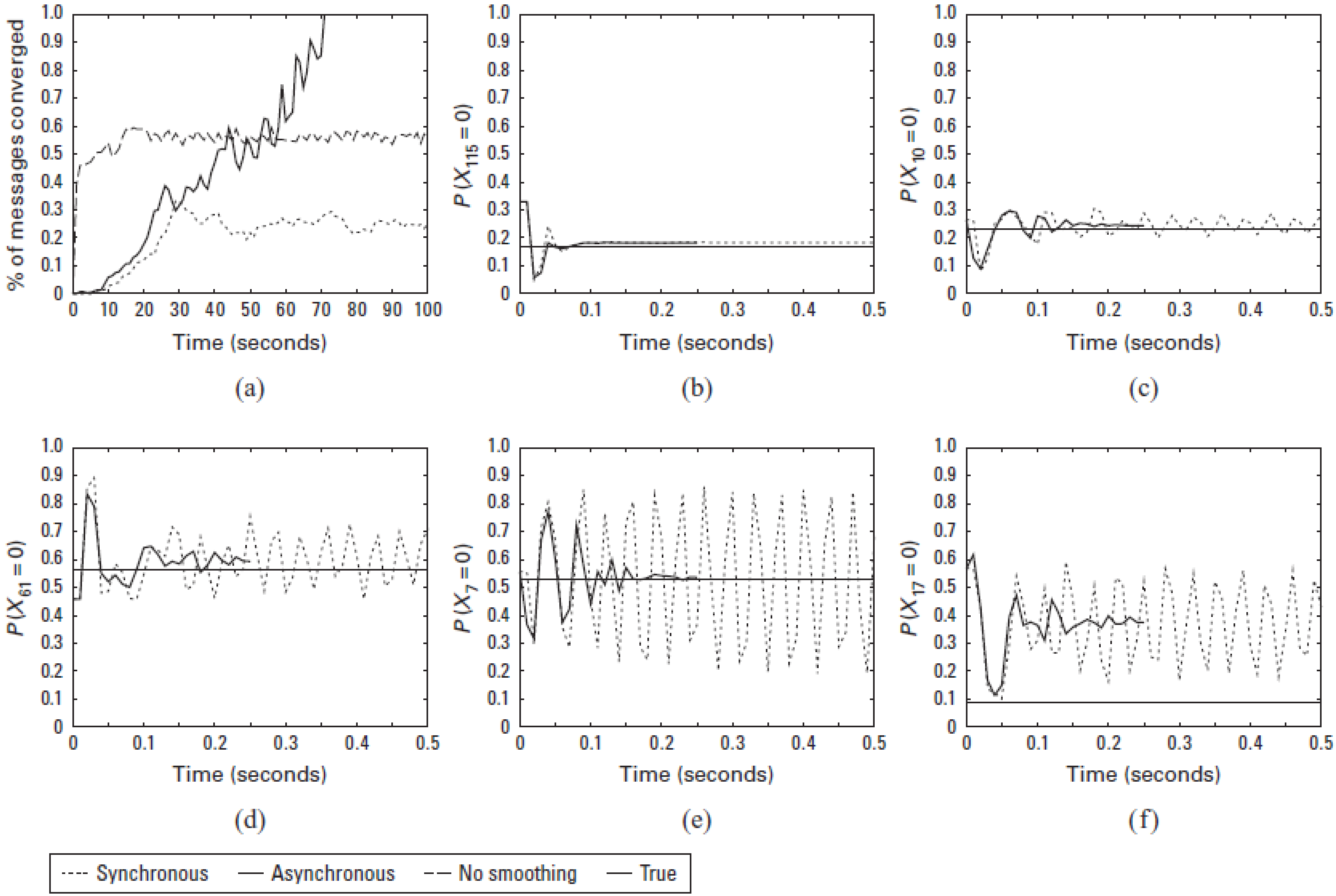
- The first concern of belief propagation is the convergence
- If converged, another concern is that the calibrated beliefs are not equal to the correct marginal potentials

Do we have some heuristic solutions?

The Problem of Convergence

- Several improvements
  - Message scheduling: residual belief propagation, order the messages according to their changes
  - Minimum spanning tree: each time select a different spanning tree of the cluster graph and do calibration
  - ......

Can we find a theoretical explanation of belief propagation on cluster graph?

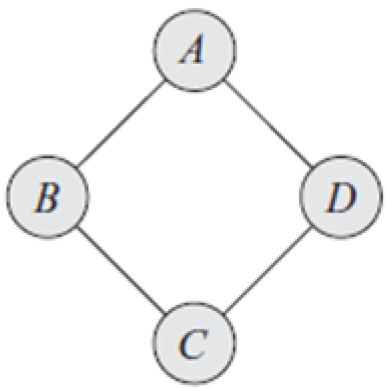


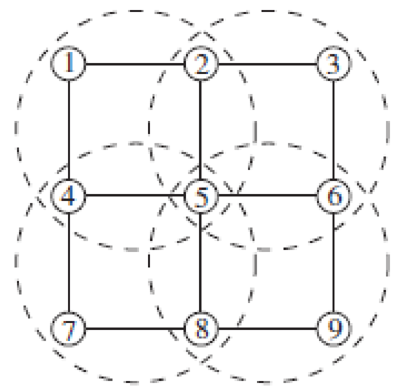
An application to a 11*11 *Ising* model

Extensions of Belief Propagation

- Generalized belief propagation
- Convex belief propagation
- Expectation propagation
- ……

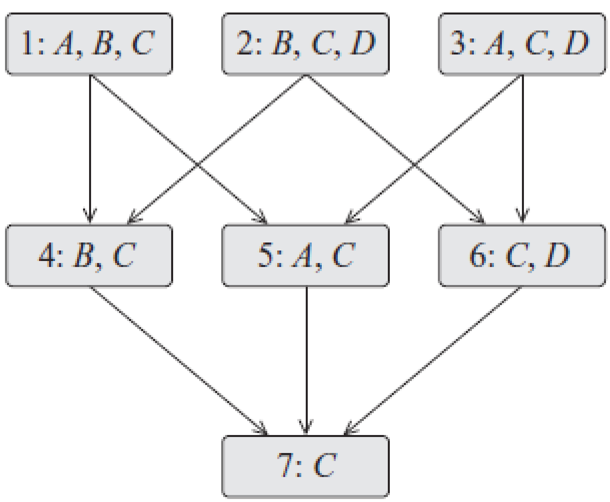
Region Graph (for Generalized BP)

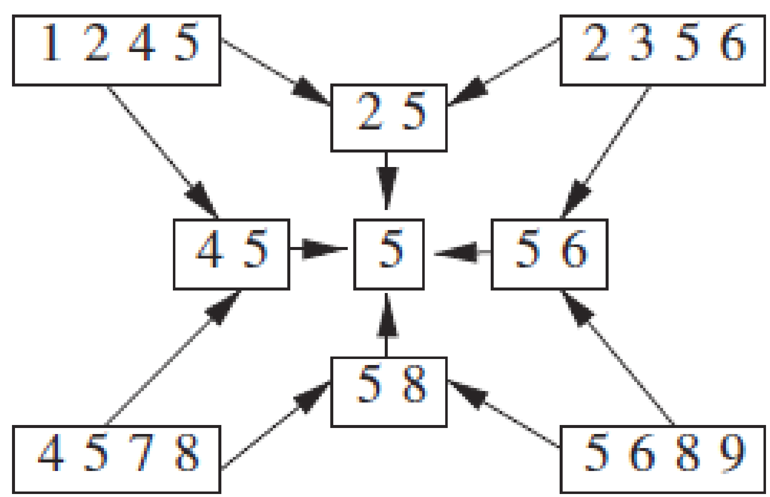




A *node* (or a *factor*) denotes a *region* (sub-graph)

The *regions* are ***hierarchically*** cut and linked by ***directed*** edges





A *factor* can be defined on any *sub-graph* rather than only *clique*

References for Belief Propagation

- Textbook #2: Chapter 22. **More variational inference**

- Wainwright MJ, Michael IJ. **Graphical models, exponential families, and variational inference**. *Foundations and Trends® in Machine Learning*, 1(1-2):1-305.

BP as Variational Inference

- The major advance of the theoretical analysis of belief propagation is that Yedidia *et al*. (2000, 2005) shows these approaches are maximizing an approximate energy functional

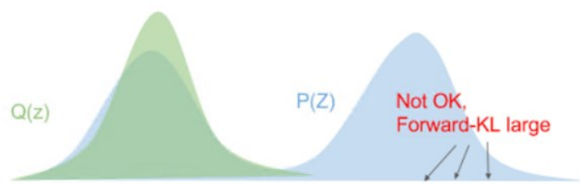
- This result connected the algorithmic developments in the field with literature on free-energy approximations developed in statistical mechanics (Bethe 1935; Kikuchi 1951), such as mean field inference

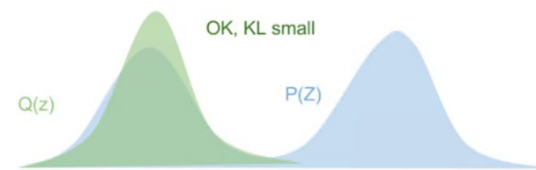
Basic Idea of Variational Inference

- To infer $P(X)$, find another distribution $Q(X)$ which can approximate it
  - Target distribution: $P(X)$
  - Proposal distribution: $Q(X)$
    - Q is restricted to a family of distribution with simple form
  - KL divergence: $\boldsymbol{D_{KL}(Q||P)}$ or $D_{KL}(P||Q)$
    - $D_{KL}(Q||P) = E_{X\sim Q}\left(\ln\left(\frac{Q}{P}\right)\right)$
  - Aim: $\min_{Q} D_{KL}(Q||P)$
- Structured variational inference
  - Define $Q(X)$ based on the graph with simple structure
    - E.g. A graph with no edge (*mean fields algorithm*)

Comments On the Two Divergences

- Assume $Q(X)$ is restricted as Gaussians
  - Forward KL (M-projection): $D_{KL}(P||Q)$
  - Backward KL (I-projection): $\boldsymbol{D_{KL}(Q||P)}$





Forward KL: Q should cover $P > 0$

Backward KL: Q should model one of the peaks in $P > 0$

Q: Why backward KL is commonly used?

Calculations in Variational Inferences

- I-projection
  - General conditional target distribution $P(X|Z{=}z)$
  - $D_{KL}(Q||P) = E_{X\sim Q}\left(\ln\frac{Q(X)}{P(X|z)}\right)$
    - $P(X|z) = P(X,z)/P(z)$
  - $D_{KL}(Q||P) = E_{X\sim Q}\left(\ln\frac{Q(X)}{P(X,z)}\right) + \ln P(z)$
  - $\ln P(z) - D_{KL}(Q||P) = -E_{X\sim Q}\left(\ln\frac{Q(X)}{P(X,z)}\right)$
  - $= E_{X\sim Q}(\ln P(X,z)) + H\big(Q(X)\big)$
- Two key questions:
  - ***Q1***: How to choose the family of the proposal distribution?
  - ***Q2***: How to maximize the above energy functional?

Exact Inference as Optimization

- Energy functional
  - $D_{KL}(Q||P_\Phi) = \ln Z - \left(H_Q(X) + \sum_{\phi\in\Phi} E_Q(\phi)\right)$
  - The second term is energy functional $F[\tilde{P}_\Phi, Q]$

- For a clique tree with clique C_i, we have a set of beliefs $\boldsymbol{Q}$ ($\beta_i, \mu_{i,j}$ - not calibrated). Its energy functional:
  - $\tilde{P}_\Phi = \prod_i \psi_i$ (ψ_i is the initial factor for each clique)
  - $\tilde{F}[\tilde{P}_\Phi, \boldsymbol{Q}] = \sum_i H_{\beta_i}(C_i) - \sum_i H_{\mu_{i,j}}\left(S_{i,j}\right) + \sum_i E_{\beta_i}[\ln \psi_i]$

Exact Inference as Optimization

- If $\boldsymbol{Q}$ is calibrated as Q^*, we can conclude that
  - $\tilde{F}[\tilde{P}_\Phi, Q^*] = \max_{\boldsymbol{Q}} \tilde{F}[\tilde{P}_\Phi, \boldsymbol{Q}] = \ln Z$
- Because the distribution is invariant for any calibrated beliefs, the relative entropy is minimized as 0
- Transform SP/BU algorithm as optimization

CTree-Optimize-KL:

Find $Q = \{\beta_i : i \in \mathcal{V}_\mathcal{T}\} \cup \{\mu_{i,j} : (i–j) \in \mathcal{E}_\mathcal{T}\}$

maximizing $-\boldsymbol{D}(Q \| P_\Phi)$

subject to

$$\mu_{i,j}[s_{i,j}] = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{c}_i) \quad \forall (i–j) \in \mathcal{E}_\mathcal{T}, \forall s_{i,j} \in Val(\boldsymbol{S}_{i,j})$$

$$\sum_{\boldsymbol{c}_i} \beta_i(\boldsymbol{c}_i) = 1 \quad \forall i \in \mathcal{V}_\mathcal{T}.$$

The End of Chapter 8

Cluster graph is another data structure for efficient inferences of PGMs