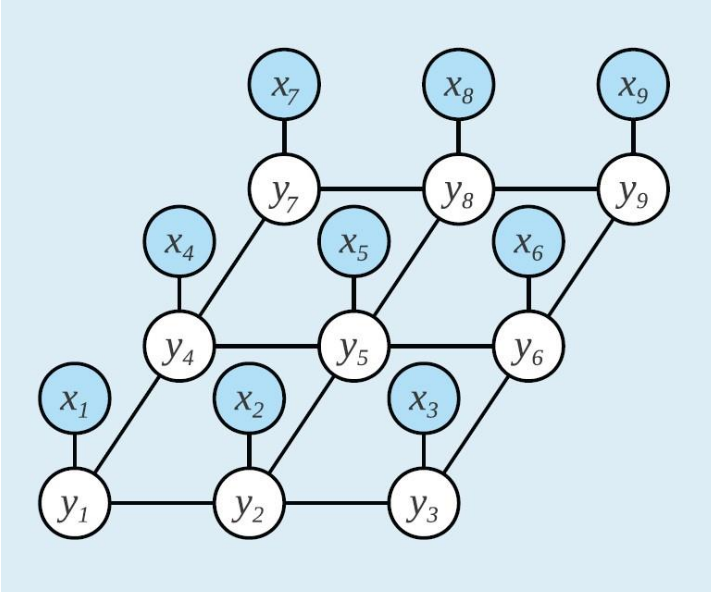
Probabilistic Graphical Models in Bioinformatics

Lecture 13: Clique trees; MAP inference



Hesam Montazeri
Department of Bioinformatics, IBB, University of Tehran

Khordad 8, 1398
(May 29, 2019)

What you have learned so far and remaining topics

University of TEHRAN

- **Representation**
  - Bayesian networks
  - Markov networks
  - Conditional independencies; conditional probability distributions; continuous and discrete data
  - Temporal models: HMM and dynamic Bayesian networks
- **Inference**
  - Variable elimination
  - Clique trees
  - MAP inference
  - Sampling methods: Gibbs sampling and MCMC
- **Learning**
  - Parameter learning: MLE, Bayesian inference
  - Structure learning: constraint-based and score-based methods
  - In the case of partially observed data: gradient ascent methods; EM
- **Applications**
  - Gene regulatory networks
  - Motif finding
  - Profile HMM for sequence alignment

Exact inference: clique tree

Exact inference: clique tree

- We already showed how variable elimination can perform exact inference effectively.
  - Key insight: performing local operations on the factors defining the distribution, rather than generating the entire joint distribution.

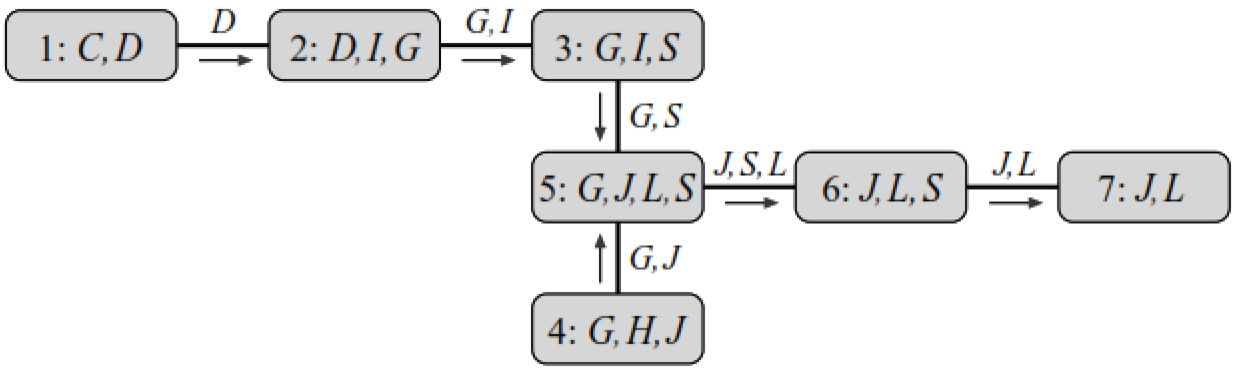
- Today we present an alternative implementation of the same insight.
  - The algorithm uses a global data structure, called clique tree, for scheduling manipulations of factors.

- In the variable elimination,
  - each step in the computation creates a factor ψ_i by multiplying existing factors.
  - A variable is them eliminated in ψ_i to generate a new factor τ_i.

- In this section, we consider a factor ψ_i to be a computational data structure
  - takes "messages" τ_j generated by other factor ψ_j
  - and generates a message τ_i used by another factor ψ_l.

Cluster tree definition

- A cluster graph U for a set of factors Φ over $\mathcal{X}$ is an undirected graph
  - each of whose nodes *i* is associated with a subset $C_i \subset \mathcal{X}$
  - each edge between a pair of clusters C_i and C_j is associated with a sepset $S_{i,j} \subseteq C_i \cap C_j$.
- A cluster graph must be family-preserving
  - Each factor $\phi \in \Phi$ must be associated with a cluster C_i

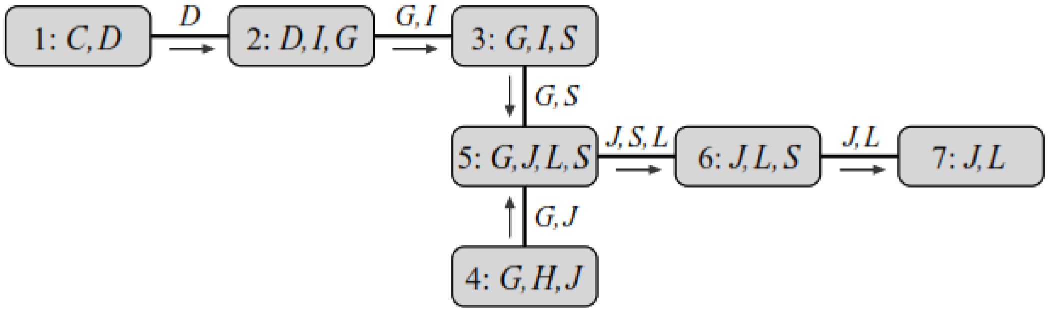


Variable elimination and cluster graph

Example revisited

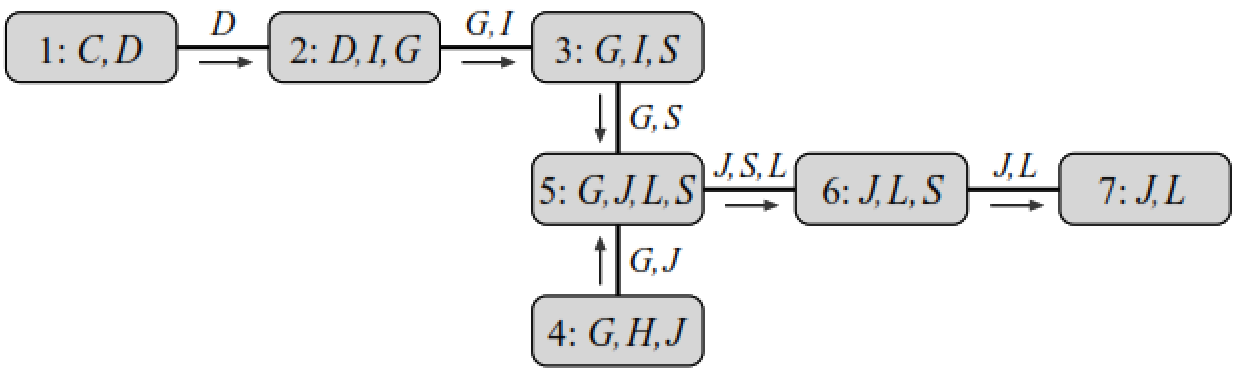
- We have seven factors $\psi_1, \ldots, \psi_7$.
- The message $\tau_1(D)$ is generated from $\psi_1(C, D)$.
- $\tau_1(D)$ participates in the computation of ψ_2.
- Hence, there is an edge between C_1 and C_2.

Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D, C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G, I, D)$, $\tau_1(D)$	G, I, D	$\tau_2(G, I)$
3	I	$\phi_I(I)$, $\phi_S(S, I)$, $\tau_2(G, I)$	G, S, I	$\tau_3(G, S)$
4	H	$\phi_H(H, G, J)$	H, G, J	$\tau_4(G, J)$
5	G	$\tau_4(G, J)$, $\tau_3(G, S)$, $\phi_L(L, G)$	G, J, L, S	$\tau_5(J, L, S)$
6	S	$\tau_5(J, L, S)$, $\phi_J(J, L, S)$	J, L, S	$\tau_6(J, L)$
7	L	$\tau_6(J, L)$	J, L	$\tau_7(J)$



Running intersection property

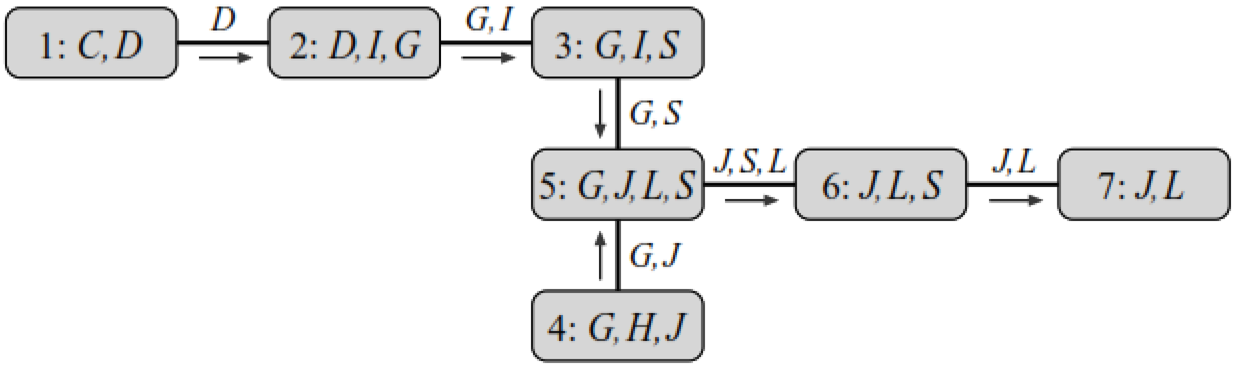
- We say a cluster tree has the running intersection property if $X \in C_i$ and $X \in C_j$, then X is also in **every** cluster in the **unique** path between C_i and C_j.

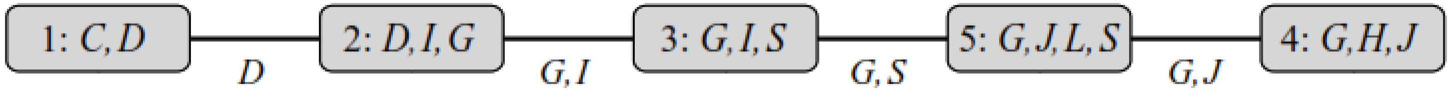


Note that running intersection property implies that $\mathrm{S}_{\mathrm{i,j}} = \mathrm{C}_i \cap C_j$.

Clique tree

- A cluster tree that satisfies the running intersection property is called a clique tree, also called a junction tree or a joint tree.

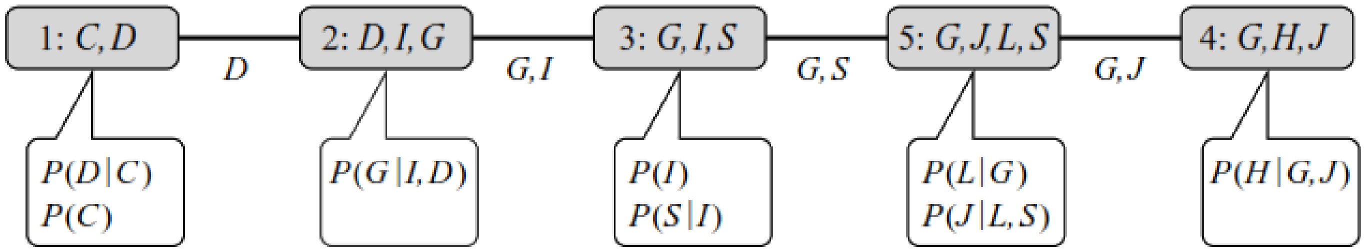




Nonmaximal cliques are absent in this simplified clique tree.

Variable elimination in a clique tree

- Consider one possible clique tree for the Student network



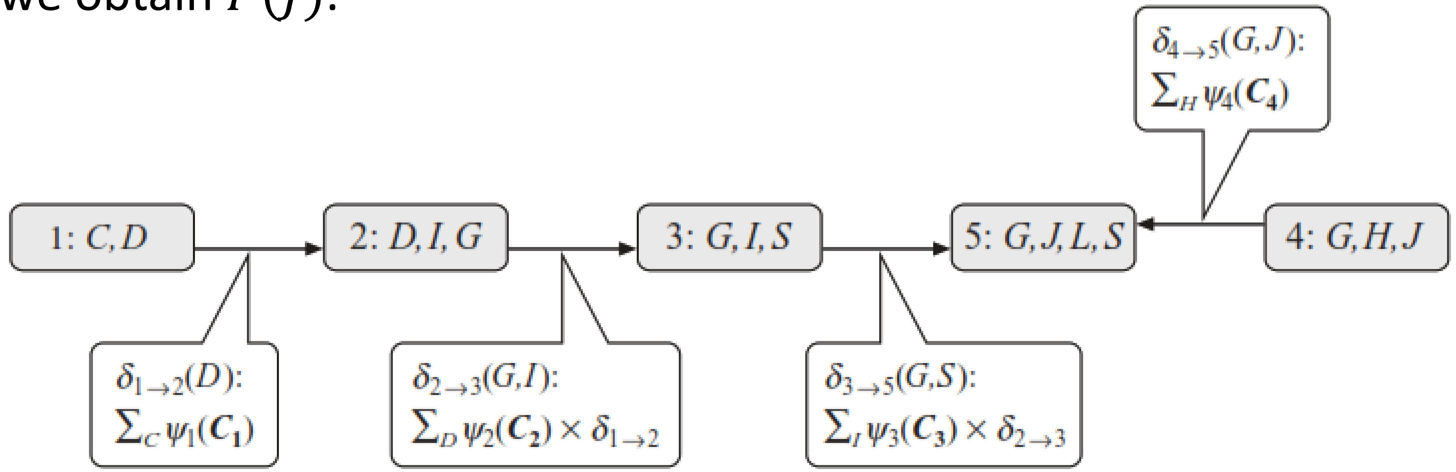
- The first step is to generate a set of *initial potentials* associated with the different cliques by multiplying the initial factors assigned each clique

$$\psi_5(J, L, G, S) = \phi_L(L, G) \cdot \phi_J(J, L, S)$$

- Assume our task is to compute $P(J)$. Then we want to do VE so the J is not eliminated.
- We select as our root clique some clique that contains J, for example C5

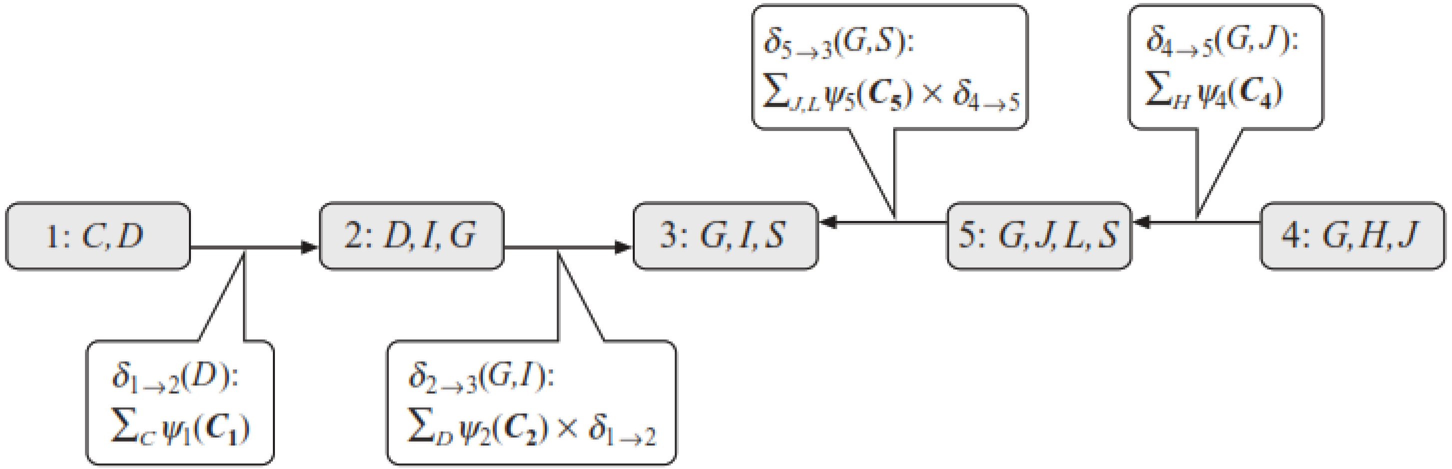
Variable elimination in a clique tree

- We then execute the following steps:
  1. In C_1: we eliminate C by performing $\sum_C \psi_1(C, D)$. The resulting factor has scope D and send it as a message $\delta_{1\to 2}(D)$ to C_2.
  2. In C_2: we eliminate D. The resulting factor has scope G, I and send it as a message $\delta_{2\to 3}(G, I)$ to C_3.
  3. In C_3: we eliminate I. The resulting factor has scope G, S and send it as a message $\delta_{3\to 5}(G, S)$ to C_5.
  4. In C_4: we eliminate H. The resulting factor has scope G, J and send it as a message $\delta_{4\to 5}(G, J)$ to C_5.
  5. In C_5: we define $\beta_5(G, J, S, L) = \delta_{3\to 5}(G, S).\, \delta_{4\to 5}(G, J).\, \psi_5(G, J, S, L)$. By summing out G, L, and S we obtain $P(J)$.



Another example: computing P(G)

- Messages sent from C_1 to C_2, from C_2 to C_3, and from C_4 to C_5 are the same as the previous run



- Ready clique: a clique is ready when it has received all of its incoming messages.

Clique-tree message passing

1. Let T be a clique tree with the cliques $C_1, \dots, C_k$.
2. We begin by multiplying the factors assigned to each clique, resulting in our initial potentials.

$$\psi_j(\boldsymbol{C}_j) = \prod_{\phi \,:\, \alpha(\phi)=j} \phi.$$

Because each factor is assigned to exactly one clique, we have

$$\prod_{\phi} \phi = \prod_{j} \psi_j.$$

3. We use clique-tree data structure to pass messages between neighboring cliques, sending all messages toward to the root clique.

$$\delta_{i\to j} = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \prod_{k\in(\mathrm{Nb}_i - \{j\})} \delta_{k\to i}.$$

4. When the root clique has received all messages, it multiplies them with its own initial potential. The result is a factor called the *beliefs*, denoted $\beta_r(C_r)$. It represents

$$\tilde{P}_\Phi(\boldsymbol{C}_r) = \sum_{\mathcal{X}-\boldsymbol{C}_r} \prod_{\phi} \phi.$$

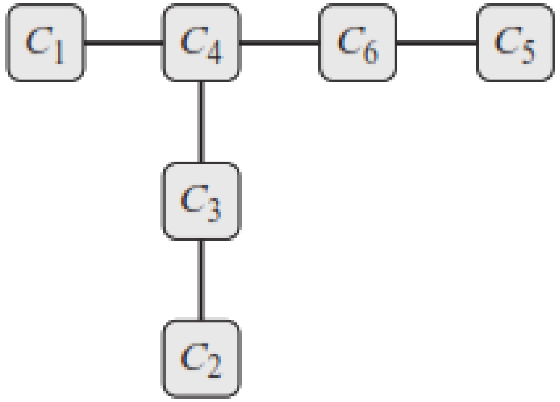
Algorithm 10.1 Upward pass of variable elimination in clique tree

Procedure CTree-SP-Upward (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  C_r // Some selected root clique
)
  Initialize-Cliques
  while C_r is not ready
    Let C_i be a ready clique
    $\delta_{i \to p_r(i)}(S_{i,p_r(i)}) \leftarrow$ SP-Message$(i, p_r(i))$
  $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \mathrm{Nb}_{C_r}} \delta_{k \to r}$
  return β_r

Procedure Initialize-Cliques (

)
  for each clique C_i
    $\psi_i(C_i) \leftarrow \prod_{\phi_j \,:\, \alpha(\phi_j)=i} \phi_j$

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

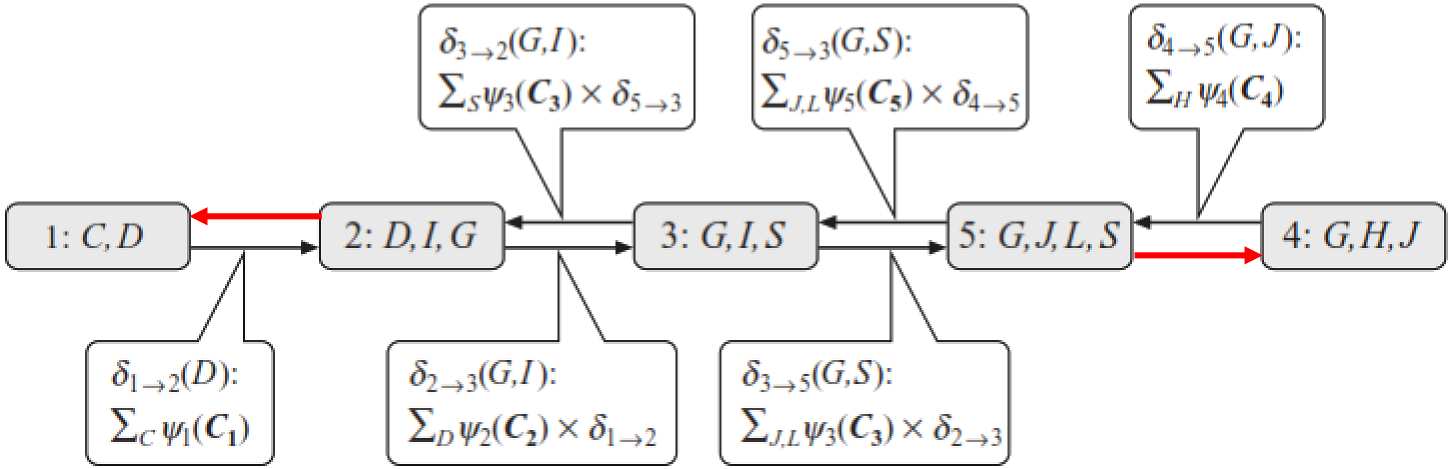


Assume we have selected C_6 as our root clique.
There are multiple legitimate orderings.

Clique tree calibration

- We showed how to use the same clique tree to compute the probability of any variable in the input variables.
- We often wish to estimate the probability of a large number of variables.
  - For example, we want to compute the probability of several possible diseases in medical diagnosis setting.
- Consider the task of computing the posterior distribution over each random variable in the network
  - Naïve approach: doing inference separately for each variable.
  - Better approach: ***sum-product belief propagation*** algorithm.

Sum-Product Belief Propagation



Sum-Product Belief Propagation

Algorithm 10.2 Calibration using sum-product message passing in a clique tree

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 $\{\beta_i\}$

Scheduling for sending messages:

- Asynchronously
- Upward pass and downward pass

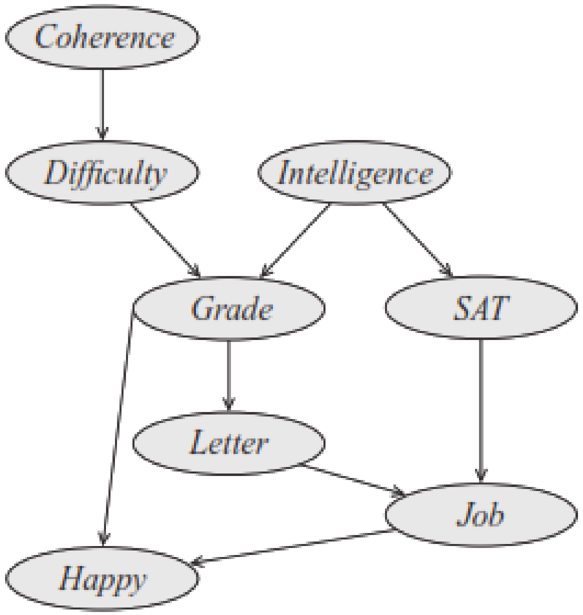
Constructing a clique tree

Constructing a clique tree

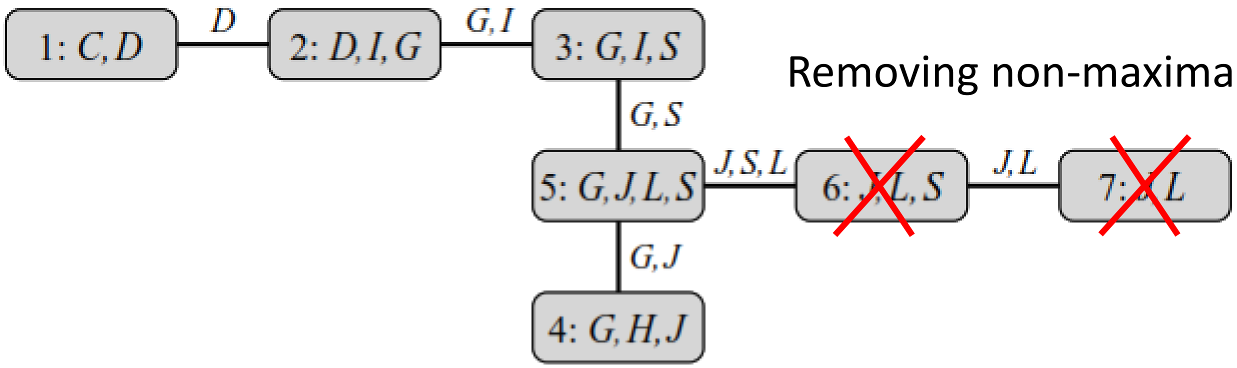
- How do we construct a clique tree for a set of factors, or equivalently, for its underlying undirected graph H_Φ? *(See section 9.4.2.1 for the definition of* H_Φ.*)*

- Two basic approaches
  - Clique trees from variable elimination
  - Clique trees from chordal graphs

Clique trees from variable elimination



Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D, C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G, I, D)$, $\tau_1(D)$	G, I, D	$\tau_2(G, I)$
3	I	$\phi_I(I)$, $\phi_S(S, I)$, $\tau_2(G, I)$	G, S, I	$\tau_3(G, S)$
4	H	$\phi_H(H, G, J)$	H, G, J	$\tau_4(G, J)$
5	G	$\tau_4(G, J)$, $\tau_3(G, S)$, $\phi_L(L, G)$	G, J, L, S	$\tau_5(J, L, S)$
6	S	$\tau_5(J, L, S)$, $\phi_J(J, L, S)$	J, L, S	$\tau_6(J, L)$
7	L	$\tau_6(J, L)$	J, L	$\tau_7(J)$

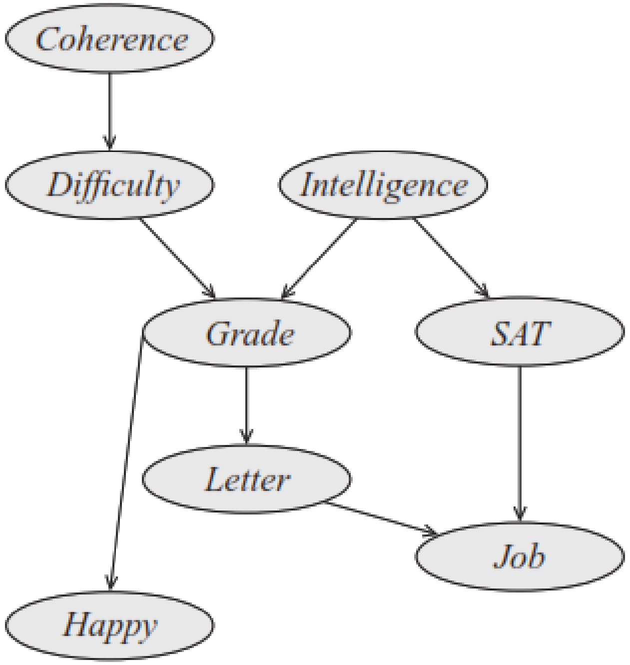


Removing non-maximal cliques

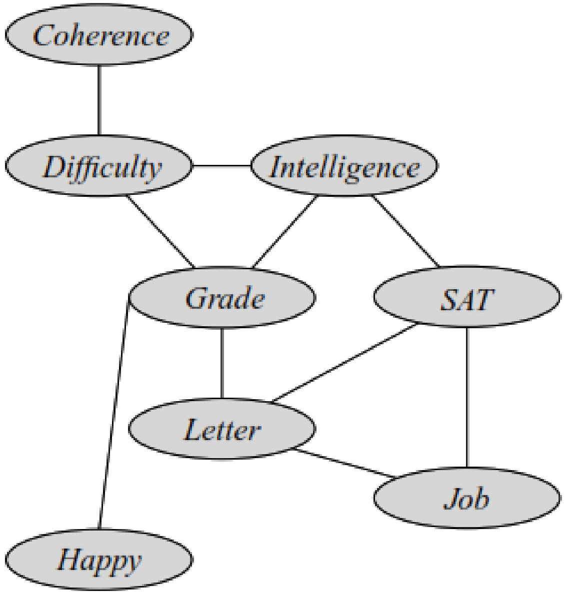
- Each cluster is defined by $C_i = \text{Scope}[\psi_i]$
- There is an edge between C_i and C_j of the message τ_i is used in computing ψ_j

Clique trees from chordal graphs

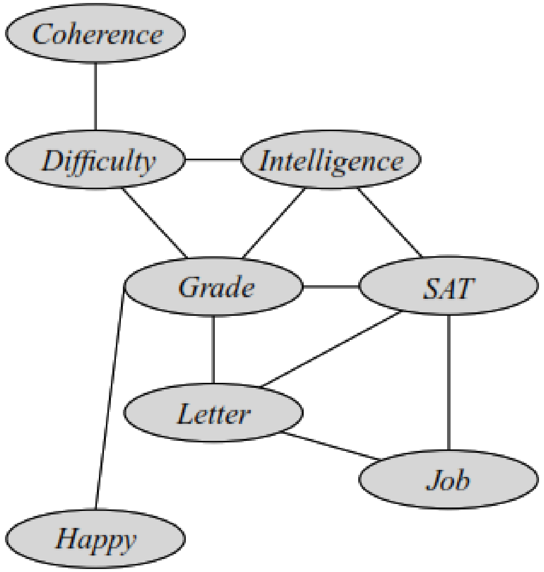
1. Find the minimal Markov network I-map H_Φ
   - For a Bayesian network, in the case without evidence, it is simply the moralized graph
2. Triangulation to obtain a chordal graph H^*
3. Find cliques in H^*, and make each one a node in the cluster graph
4. Run the maximum spanning tree algorithm on the cluster graph to construct a tree

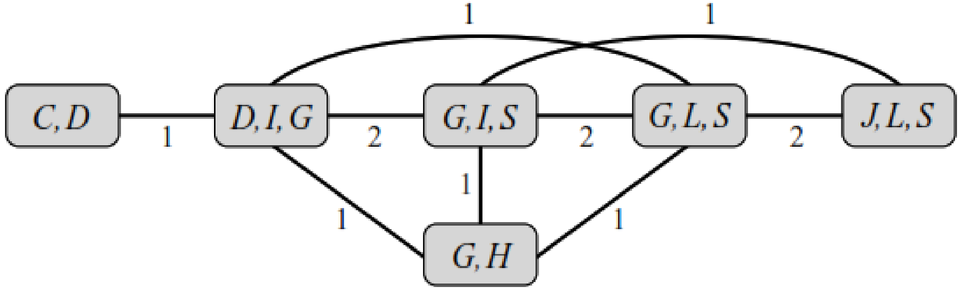


moralization

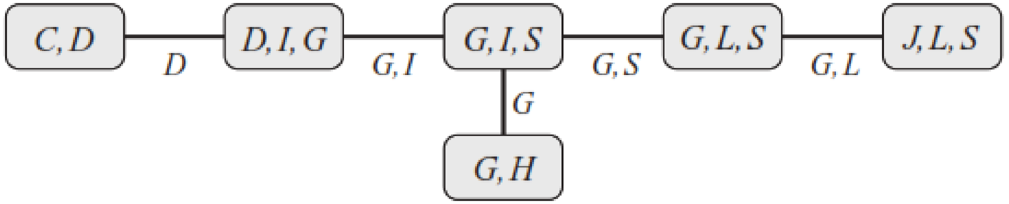


triangulation





Maximum spanning tree



MAP inference

MAP inference

- A MAP query aims to find the most likely assignment to all of the (non-evidence) variables.

- A marginal MAP query aims to find the most likely assignment to a subset of the variables, marginalizing over the rest.

- Examples:
  - Speech recognition: to decode the most likely utterance given the (noisy) acoustic signal.

- For addressing MAP queries, we need to compute

$$\xi^{map} = \arg\max_{\xi} P_{\Phi}(\xi) = \arg\max_{\xi} \frac{1}{Z} \tilde{P}_{\Phi}(\xi) = \arg\max_{\xi} \tilde{P}_{\Phi}(\xi).$$

Max-Product Variable Elimination

- Consider the Bayesian network $A \rightarrow B$. Assume our goal is to compute:

$$\begin{aligned}\max_{a,b} P(a,b) &= \max_{a,b} P(a)P(b \mid a) \\ &= \max_{a} \max_{b} P(a)P(b \mid a).\end{aligned}$$

- For any choice of a, the value of B must be chosen so as to maximize $P(b \mid a)$. We must also choose the value of A appropriately

$$\max_{b} P(a)P(b \mid a) = P(a) \max_{b} P(b \mid a).$$

- Let $\phi(a)$ denote $\max_{b} P(b \mid a)$. For example, for the following assignments

a^0	a^1
0.4	0.6

A	b^0	b^1
a^0	0.1	0.9
a^1	0.55	0.45.

- $\phi(a^1) = \max_{b} P(b \mid a^1) = 0.55$ and $\phi(a^0) = \max_{b} P(b \mid a^0) = 0.9$. Finally

$$\max_{a} P(a)\phi(a) = \max\,[0.4 \cdot 0.9, 0.6 \cdot 0.55] = 0.36.$$

Max-marginalization

Let $\phi(X, Y)$ be a factor. We define the factor maximization of Y in ϕ as

$$\psi(\boldsymbol{X}) = \max_{Y} \phi(\boldsymbol{X}, Y).$$

a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

a^1	c^1	0.25
a^1	c^2	0.35
a^2	c^1	0.05
a^2	c^2	0.07
a^3	c^1	0.15
a^3	c^2	0.21

Algorithm 13.1 Variable elimination algorithm for MAP. The algorithm can be used both in its max-product form, as shown, or in its max-sum form, replacing factor product with factor addition.

Procedure Max-Product-VE (
  Φ, // Set of factors over $\boldsymbol{X}$
  $\prec$ // Ordering on $\boldsymbol{X}$
)
  Let $X_1, \ldots, X_k$ be an ordering of $\boldsymbol{X}$ such that
    $X_i \prec X_j$ iff $i < j$
  for $i = 1, \ldots, k$
    $(\Phi, \phi_{X_i}) \leftarrow$ Max-Product-Eliminate-Var(Φ, X_i)
  $\boldsymbol{x}^* \leftarrow$ Traceback-MAP($\{\phi_{X_i} : i = 1, \ldots, k\}$)
  return $\boldsymbol{x}^*, \Phi$ // Φ contains the probability of the MAP

Procedure Max-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)
  $\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
  $\Phi'' \leftarrow \Phi - \Phi'$
  $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
  $\tau \leftarrow \max_Z \psi$
  return $(\Phi'' \cup \{\tau\}, \psi)$

Procedure Traceback-MAP (
  $\{\phi_{X_i} : i = 1, \ldots, k\}$
)
  for $i = k, \ldots, 1$
    $\boldsymbol{u}_i \leftarrow (x^*_{i+1}, \ldots, x^*_k)\langle Scope[\phi_{X_i}] - \{X_i\}\rangle$
      // The maximizing assignment to the variables eliminated after X_i
    $x^*_i \leftarrow \arg\max_{x_i} \phi_{X_i}(x_i, \boldsymbol{u}_i)$
      // x^*_i is chosen so as to maximize the corresponding entry in the factor, relative to the previous choices $\boldsymbol{u}_i$
  return $\boldsymbol{x}^*$

Theorem 13.4 *The algorithm of algorithm 13.1 returns*

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} \prod_{\phi \in \Phi} \phi,$$

and Φ, which contains a single factor of empty scope whose value is:

$$\max_{\boldsymbol{x}} \prod_{\phi \in \Phi} \phi.$$

A run of max-product variable elimination

Step	Variable eliminated	Factors used	Intermediate factor	New factor
1	S	$\phi_S(I,S)$	$\psi_1(I,S)$	$\tau_1(I)$
2	I	$\phi_I(I)$, $\phi_G(G,I,D)$, $\tau_1(I)$	$\psi_2(G,I,D)$	$\tau_2(G,D)$
3	D	$\phi_D(D)$, $\tau_2(G,D)$	$\psi_3(G,D)$	$\tau_3(G)$
4	L	$\phi_L(L,G)$	$\psi_4(L,G)$	$\tau_4(G)$
5	G	$\tau_4(G)$, $\tau_3(G)$	$\psi_5(G)$	$\tau_5(\emptyset)$

For example, the first step would compute

$$\tau_1(I) = \max_s \phi_S(I,s)$$

The final factor, $\tau_5(0)$, is simply a number, whose value is

$$\max_{S,I,D,L,G} P(S,I,D,L,G).$$

Decoding or finding the most probable assignment

- Traceback of the solution
- We begin by computing g

Step	Variable eliminated	Factors used	Intermediate factor	New factor
1	S	$\phi_S(I, S)$	$\psi_1(I, S)$	$\tau_1(I)$
2	I	$\phi_I(I)$, $\phi_G(G, I, D)$, $\tau_1(I)$	$\psi_2(G, I, D)$	$\tau_2(G, D)$
3	D	$\phi_D(D)$, $\tau_2(G, D)$	$\psi_3(G, D)$	$\tau_3(G)$
4	L	$\phi_L(L, G)$	$\psi_4(L, G)$	$\tau_4(G)$
5	G	$\tau_4(G)$, $\tau_3(G)$	$\psi_5(G)$	$\tau_5(\emptyset)$

$$g^* = \arg\max_g \psi_5(g)$$

$$l^* = \arg\max_l \psi_4(g^*, l)$$

$$\begin{aligned} d^* &= \arg\max_d \psi_3(g^*, d) \\ i^* &= \arg\max_i \psi_2(g^*, i, d^*) \\ s^* &= \arg\max_s \psi_1(i^*, s). \end{aligned}$$

(Refer to Example 13.4 for more details)