Graphical Models for Inference and Decision Making

Unit 4: Inference in Graphical Models
The Junction Tree Algorithm

Instructor: Kathryn Blackmond Laskey

Spring 2019

Learning Objectives

1. State the canonical belief propagation problem for inference in probabilistic graphical models.
2. Describe why inference is hard for general graphical models.
3. Explain how the junction tree inference algorithm works.
   a) Given a Bayesian network, construct a junction tree.
   b) Given a junction tree, follow the steps for propagating beliefs between two neighboring clusters.
   c) Given a junction tree, describe the flow of control as belief propagation occurs.
4. Describe how the junction tree algorithm is modified for:
   a) Virtual evidence
   b) Most probable explanation (MPE) or maximum a posteriori (MAP) inference

Unit 4 Outline

- The Inference Problem
- Pearl's Belief Propagation Algorithm
- Junction Tree Algorithm
  - Propagation in Junction Trees
  - Constructing the Junction Tree
  - Additional topics:
    - » Virtual (likelihood) evidence & soft evidence
    - » MAP / MPE

Inference in Graphical Probability Models

- In a graphical probability model the joint probability distribution over all random variables in the network is encoded by specifying
  - the graph
  - the local probability distributions
- Inference (evidence accumulation, belief propagation) means using new information about some random variables to update beliefs about other random variables
- Incorporate new information by computing conditional distributions
  - Given:
    - » Target random variables X_t; evidence random variables X_e; other random variables X_o
    - » A graphical probability model on (X_t, X_e, X_o)
    - » Evidence about X_e (either we learn $X_e = x_e$ or we receive "virtual evidence")
  - Goal:
    - » Answer queries about conditional distribution $P(X_t \mid X_e = x_e)$ or $P(X_t \mid \text{virtual evidence})$
    - » Assume that we want to know the distribution of one of the target random variables (we will discuss other kinds of query later)
- No need to specify in advance which random variables are target / evidence
  - A graphical model encodes a full joint distribution over all the random variables
  - The same graphical model can be used to infer fever from flu and flu from fever ("bidirectional inference")

MPE and MAP

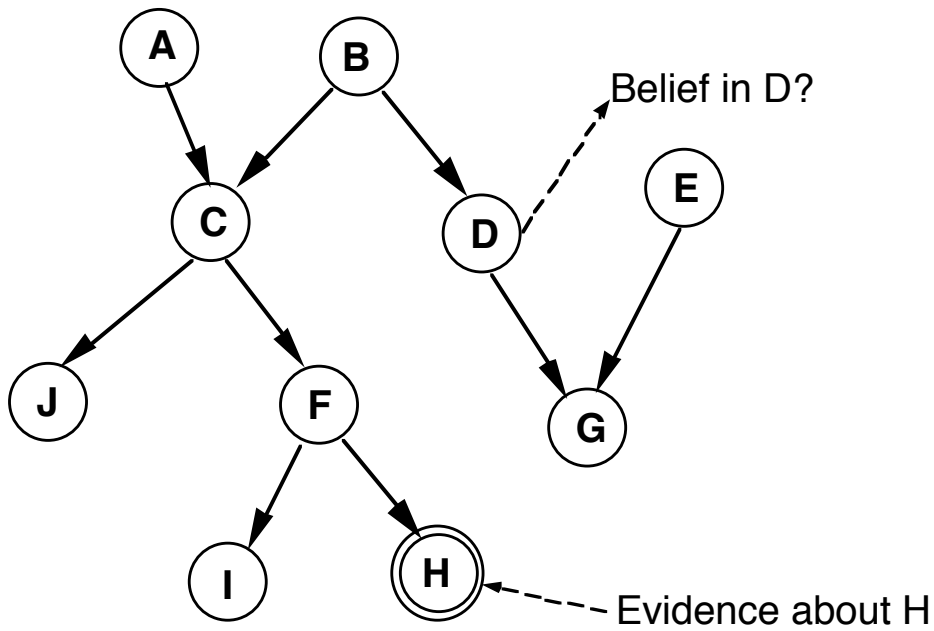
- BNs are also used to solve the maximum *a posteriori* (MAP) or most probable explanation (MPE) problem:
  - MPE: Given evidence $X_e=x_e$, find the most probable instantiation of the non-evidence random variables (X_t,X_o)
  - MAP: Given evidence $X_e=x_e$, find the most probable instantiation of a target random variable X_t
  - *The answer to the second problem is not the same as the answer to the first!!! Why?*
- We will focus on the computation of posterior marginal distributions for target random variables
  - Computational issues for MAP and MPE are similar but not identical to issues for posterior marginal distributions
  - Published literature on MAP and MPE should be understandable to someone who thoroughly understands algorithms for computing posterior marginal distributions
  - We will briefly describe how to modify the algorithm to find MPE

Pearl on Belief Updating

- "The primary appeal of probability theory is its ability to express qualitative relationships among beliefs and to process these relationships in a way that yields intuitively plausible conclusions."
- "The fortunate match between human intuition and the laws of proportions is not a coincidence. It came about because beliefs are not formed in a vacuum but rather as a distillation of sensory experiences. For reasons of storage economy and generality we forget the actual experiences and retain their mental impressions in the forms of averages, weights, or (more vividly) abstract qualitative relationships that help us determine future actions."
- "We [view] a belief network not merely as a passive code for storing factual knowledge but also as a computational architecture for reasoning about that knowledge. This means that the links in the network should be treated as the only mechanisms that direct and propel the flow of data through the process of querying and updating beliefs."
  - exploits independencies in network
  - leads to conceptually meaningful belief revision process
  - separates control from knowledge representation
  - natural object-oriented implementation
- Probability updating seems to resist local constraint propagation approaches because of the need to keep track of sources of belief change.
- Purely local and distributed updating is possible in singly connected networks.

The Inference (aka "belief propagation") Problem

- Functions the inference algorithm performs:
  - Query a node for its current probabilities
  - Declare evidence about nodes in the network
  - Update current probabilities for nodes given evidence



- Updating should conform to Bayes Rule. That is, if we learn that $X_e=x_e$, the updated belief in X_t must satisfy

$$P(X_t = x_t \mid X_e = x_e) = \frac{P(X_t = x_t, X_e = x_e)}{P(X_e = x_e)}$$

Some General Points on Inference in Graphical Models

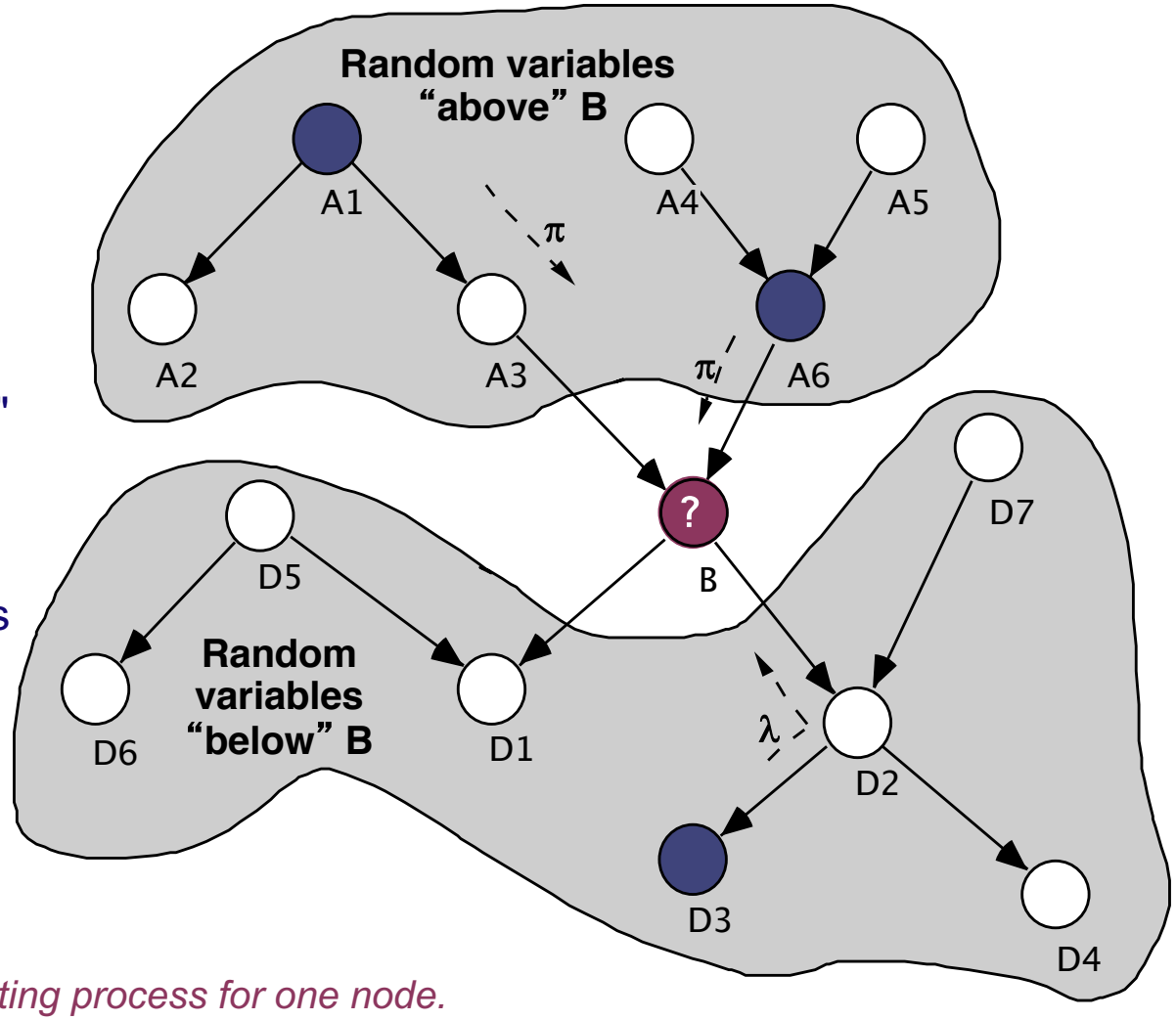
- Inference is **NP-hard** (Cooper, 1988)
  - **NP** is a class of decision problems for which any instances with "yes" answers have proofs verifiable in polynomial time (but not necessarily provable in polynomial time)
  - **NP-hard** is a class of problems at least as hard as the hardest problems in NP
    - » Formally, H is NP-hard if any problem in NP can be reduced in polynomial time to H
  - If any NP-hard problem could be solved in polynomial time than any problem in NP could also
  - No such algorithm has yet been found, and most people believe there is none
- Although general inference is intractable, tractable algorithms are available for certain types of networks
  - Singly connected networks
  - Some sparsely connected networks
  - Networks amenable to various approximation algorithms
- Goal of the expert system builder: find a representation which is a good approximation to the expert's probability distribution and has good computational properties
- Goal of the machine learning algorithm designer: search over the space of tractable models for a model which is a good approximation to the training sample
- Goal of the inference algorithm designer: find an algorithm that is tractable and provides an accurate approximation to the query of interest

Unit 4 Outline

- The Inference Problem
- Pearl's Belief Propagation Algorithm
- Junction Tree Algorithm
  - Propagation in Junction Trees
  - Constructing the Junction Tree
  - Additional topics:
    - » Virtual (likelihood) evidence & soft evidence
    - » MAP / MPE

Pearl's Algorithm for Singly Connected BNs

- Goal: compute probability distribution of random variable B given evidence (assume B itself is not known)
- Key idea: impact of belief in B from evidence "above" B and evidence "below" B can be processed separately
- Justification: B *d*-separates "above" random variables from "below" random variables



- *This picture depicts the updating process for one node.*
- *The algorithm simultaneously updates beliefs for all the nodes.*
- *Can you find the "above" and "below" subsets for other nodes in this diagram?*

= evidence random variable

Pearl's Algorithm (continued)

- Partition evidence random variables into:
  - D = evidence variables "below" B
  - A = evidence variables "above" B
- General formula for joint probability:
  - P(A,B,D) = P(A)P(B|A)P(D|B,A)
- In a singly connected network:
  - A is independent of D given B
  - Therefore P(A,B,D) = P(A)P(B|A)P(D|B)
  - From Bayes Rule we know P(B | A=a,D=d) is proportional to P(A=a, B, D=d)
  - P(B | A=a,D=d) $\propto$ P(B | A=a)P(D=d | B)
- The algorithm maintains for each node
  - π(B) = P(B | A=a) (a is the instantiated value of A)
  - λ(B) = P(D=d | B) (d is the instantiated value of D)
  - P'(B) = P(B | A=a,D=d) $\propto$ π(B)λ(B)
- When new evidence comes in:
  - If evidence is from above a π-message is sent to update π(B)
  - If evidence is from below a λ-message is sent to update λ(B)
  - P'(B) is recomputed

X is "below" B is there is a chain between it and B that goes through a child of B
- *Descendents of B*
- *Other ancestors of descendents of B*

X is "above B if it is not "below" B
- *Ancestors of B*
- *Other descendents of ancestors of B*

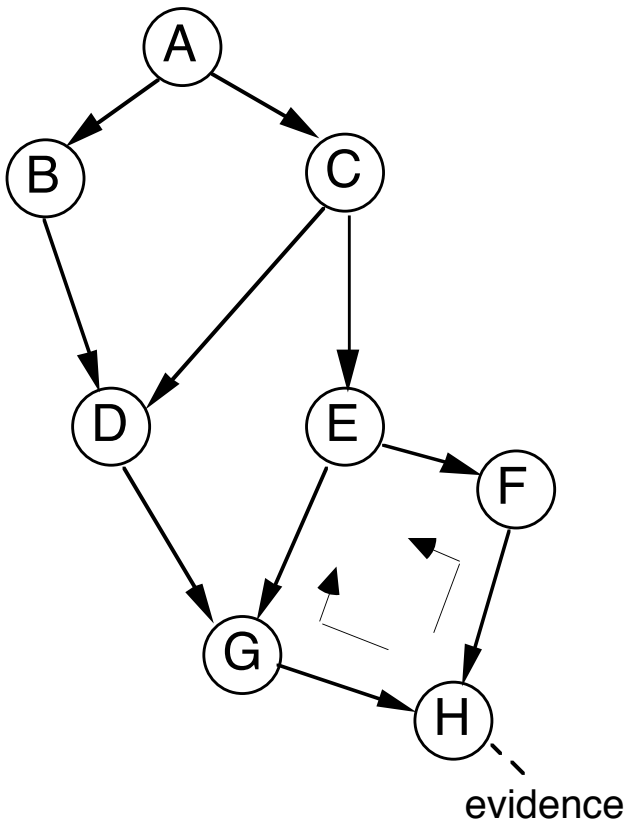
The partitioning into "above" and "below" cannot be done if the network is not singly connected

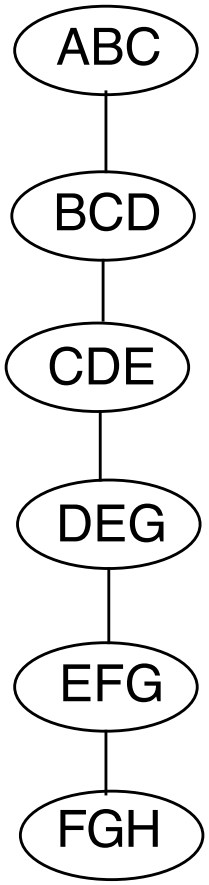
Unit 4 Outline

- The Inference Problem
- Pearl's Belief Propagation Algorithm
- Junction Tree Algorithm
  - Propagation in Junction Trees
  - Constructing the Junction Tree
  - Additional topics:
    - » Virtual (likelihood) evidence & soft evidence
    - » MAP / MPE

Multiply Connected Networks

- In a multiply connected network evidence can reach a node via multiple paths
  - Pearl's algorithm may never converge
  - No way for local propagation to account for the correlation induced by multiple evidential pathways
- Solution (if network is not too densely connected)
  - Combine nodes into clusters
  - Propagate beliefs on clusters of nodes
  - Project resulting beliefs back to the variables in the initial network
- The junction tree algorithm is a principled way to do this





Overview of Junction Tree Propagation

- The basic idea is to transform the graph into clusters of nodes so that the graph of clusters is singly connected and has the *junction tree property* (this property ensures that evidence propagates correctly)
- The junction tree becomes a permanent part of the knowledge representation, and changes only if the graph changes
- Constructing a junction tree from a graphical probability model is inherently non-modular, but the junction tree itself is a modular representation
- Inference is performed in the junction tree using a local message-passing algorithm
- Each cluster of nodes maintains its local belief table which can be used to calculate response to query for current beliefs of nodes in the cluster

Steps in Junction Tree Propagation

1. Transform BN or MN into junction tree
   - Permanent part of the knowledge representation
   - Changes only if the graph changes

   Defer until later

2. Use local local message-passing algorithm to propagate beliefs in the junction tree
   - Each cluster and each separator has a local belief table
   - Local belief table is proportional to joint probability of nodes in cluster / separator
   - Product of cluster tables divided by product of separator tables is proportional to joint probability distribution or BN
3. Query on any node or any set of nodes in same cluster can be computed from cluster table

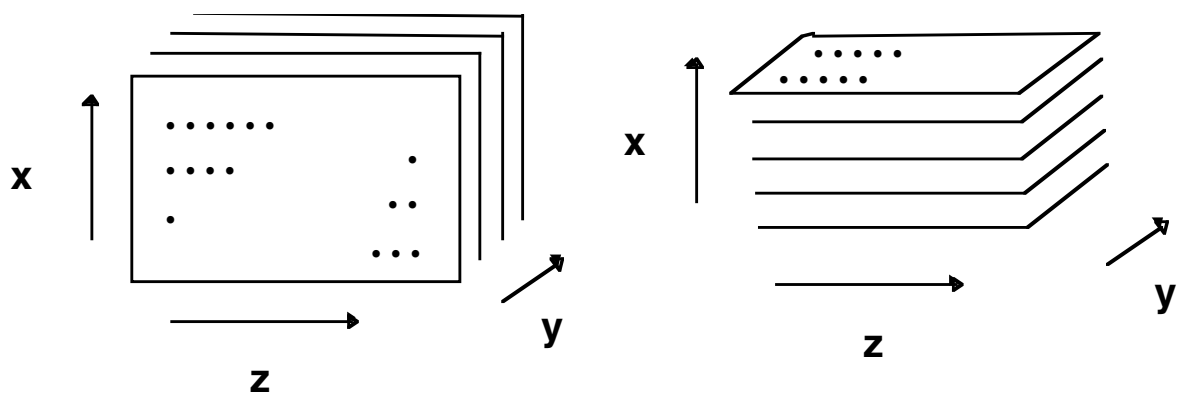
Coming next

Unit 4 Outline

- The Inference Problem
- Pearl's Belief Propagation Algorithm
- Junction Tree Algorithm
  - Propagation in Junction Trees
  - Constructing the Junction Tree
  - Additional topics:
    - » Virtual (likelihood) evidence & soft evidence
    - » MAP / MPE

Belief Tables

- A belief table on a "universe" $U=\{U_1, \ldots, U_k\}$ (cluster of nodes) is a k-dimensional array of nonnegative numbers, one for each combination of values of nodes in U.
  - Belief tables are used to represent joint probability distributions over U
- Operations on belief tables:
  - Multiplication (entry by entry)
  - Division (entry by entry)
  - Marginalization (summing out some variables in U)
  - Normalization (dividing all entries by the sum of all entries)
- To multiply and divide belief tables over different universes:
  - Extend each table by making a copy for each value of the missing variables
  - Multiply or divide entry by entry



Belief Tables: Formal Definitions

- D4.1: Let U be a universe of RVs $U=\{U_1,\ldots, U_k\}$. A belief table T_U on U is a set of nonnegative numbers, one for each combination of possible values of RVs in U. For each $u=\{u_1,\ldots, u_k\}$, we write the corresponding entry as $T_U(u)$.
- D4.2: A belief table on U is a joint probability table if all entries sum to 1.
- D4.3: Let $U= X\cup Z$ and $V= Y\cup Z$ be two universes of RVs, where X,Y, and Z are disjoint. Let T_U and T_V be belief tables on U and V, respectively. Let $W=U\cup V= X\cup Y\cup Z$ be the union of the two universes. We can write w=(x,y,z) for each value w of W. The product $T_W = T_U\times T_V$ of the belief tables is the table constructed by setting $T_W(x,y,z) = T_U(x,z)T_V(y,z)$.
- D4.4: Let $U= X\cup Z$ and $V= Y\cup Z$ be two universes of variables, where X,Y, and Z are disjoint. Let T_U and T_V be belief tables on U and V, respectively. Let $W=U\cup V= X\cup Y\cup Z$ be the union of the two universes. We can write w=(x,y,z) for each value w of W. The quotient $T_W = T_U/T_V$ of the belief tables is the table constructed by setting $T_W(x,y,z) = T_U(x,z)/T_V(y,z)$ when $T_V(y,z) \neq 0$ or $T_U(x,z) = 0$ and $T_V(y,z) = 0$. The quotient is undefined if $T_V(y,z)=0$ and $T_U(x,z) \neq 0$.
- D4.5: Let $W= U\cup V$ be a universe of variables where U and V are disjoint. Let T_W be a belief table on W. The marginal belief table of T_W over V is a belief table on U given by:

$$T_U(u) = \sum_V T_W(u,v)$$

More Formal Definitions

- D4.6: Let T_U be a belief table on U such that the sum of the entries in T_U is positive (i.e., not all entries in T_U are zero). The *normalized belief table* T'_U is the belief table on U given by dividing each entry in T_U by the sum of the entries in T_U. If all entries in T_U are zero, the normalized belief table is undefined.
  - Note: A normalized belief table on U is a joint probability table.
- D4.7: A *finding* on a random variable V is a belief table on V in which all entries are zero or one.
  - Semantically, a finding corresponds to learning that the values with zero entries are impossible.
- D4.8: Let T_U be a belief table and let $F_1,\ldots,F_m$ be findings. The conditional belief table $T_U|F_1,\ldots,F_m$ is the belief table obtained by normalizing the product $T_U \times F_1 \times \cdots \times F_m$.

- <u>Theorem 4.1</u>: If T_U is a joint probability table and $F_1,\ldots,F_m$ are findings, then the conditional belief table $T_U|F_1,\ldots,F_m$ is the probability distribution T_U conditioned on the values for which no F_i has a zero entry.

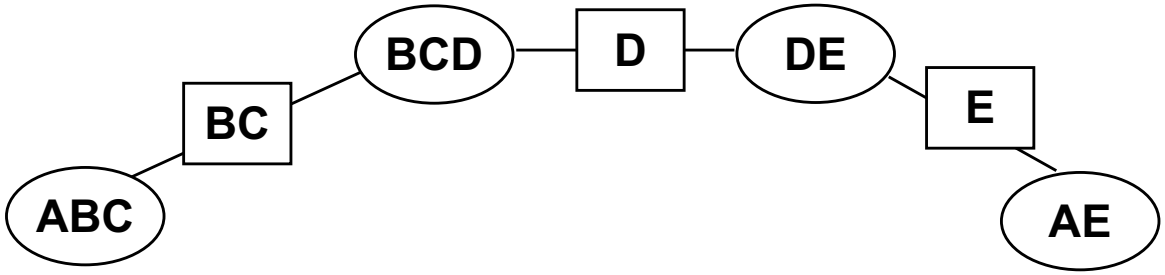
Belief Tables and Belief Propagation

- We could perform belief propagation by creating a single large belief table over all random variables in graphical model, and updating that table when evidence is received.
  - Each finding would be a belief table that had a 1 for all values consistent with the new evidence and a zero for all values not consistent with the new evidence
- This is of course intractable when there are more than a few random variables
- For not-too-densely connected networks we can cluster the random variables and arrange the clusters in a junction tree. Inference is performed by:
  - Maintaining local belief tables for each cluster in the junction tree so that the cluster belief table is a joint probability table for the nodes in the cluster
  - Passing messages between clusters when evidence arrives to update the local belief tables to incorporate the new evidence

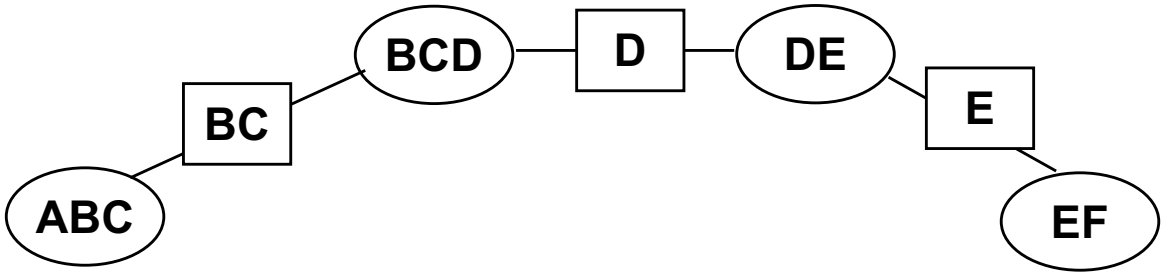
Cluster Trees and Junction Trees

- D4.9: Let U and V be two universes of RVs. The intersection $U \cap V$ is called the *separator* for U and V. (Note that U and V are conditionally independent given their separators.)
- D4.10: A *cluster tree* over U is an undirected graph G in which:
  - G is a tree (there is at most only one path between any two nodes);
  - The nodes of G are subsets of U (called *clusters*);
  - The union of the subsets for all the nodes is U;
  - Each arc between two nodes V and W is labeled with the separator $V \cap W$;
  - Associated with each node and each separator is a belief table over the random variables in the node or separator
- D4.11: A cluster tree is a junction tree if for each pair of nodes V, W, all nodes on the path from V to W contain the intersection $V \cap W$.
  - This condition is called the *junction tree property*
- D4.12: A junction tree *corresponds to* a probability distribution P(U) over U if P(U) is the product of all the cluster tables divided by the product of all the separator tables.

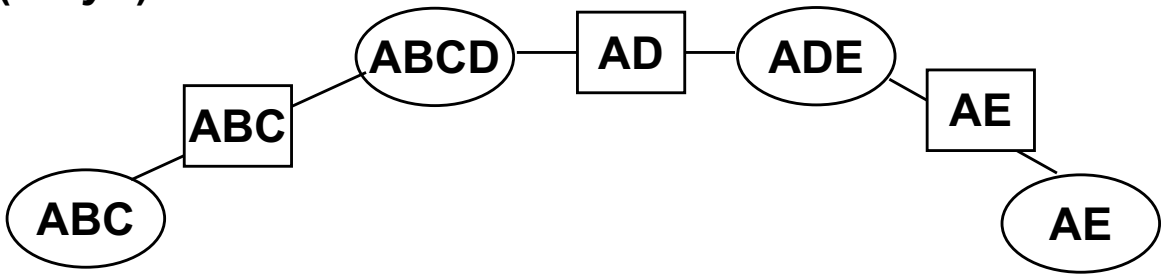
Examples

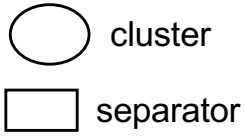


A cluster tree that is not a junction tree (*Why not?*)



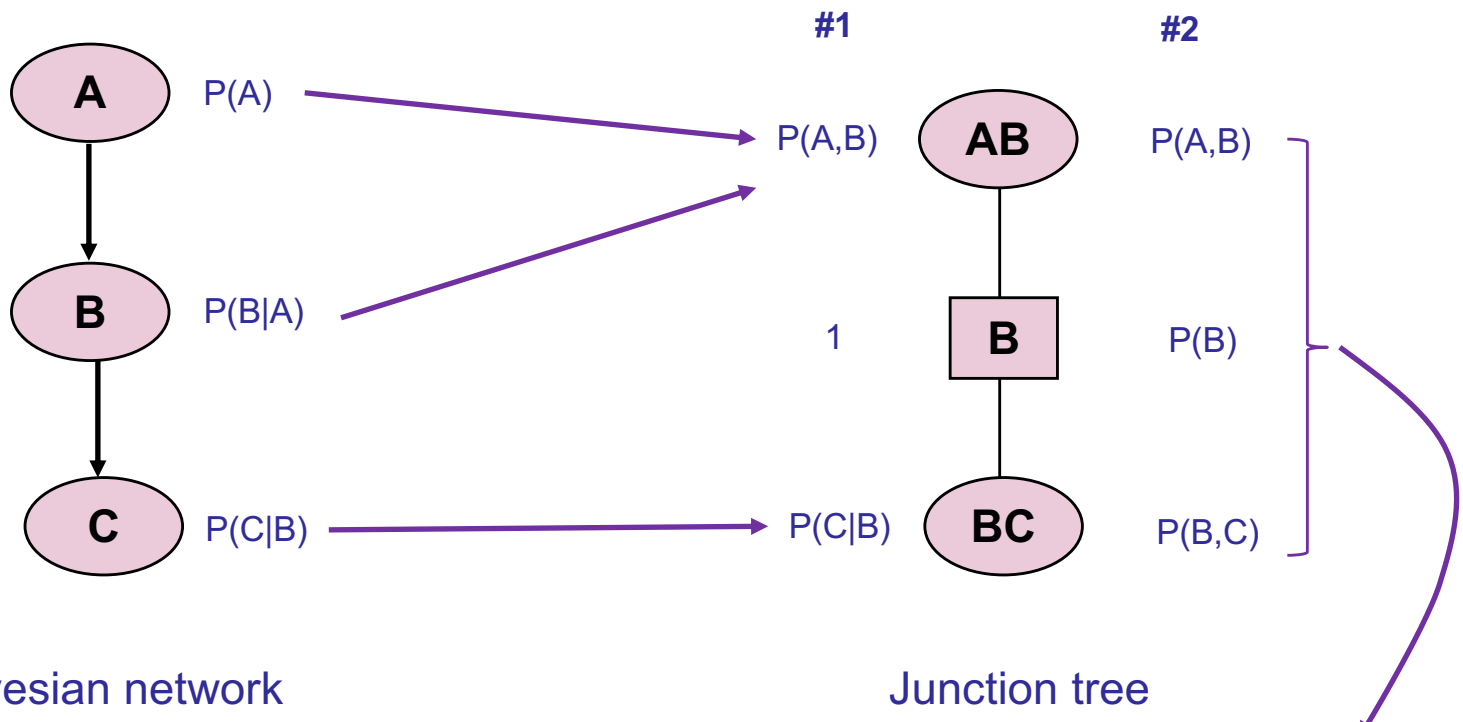
A junction tree (*Why?*)





Also a junction tree (*Why?*)

Example: Junction Tree Corresponding to a Bayesian Network



$$\frac{P(A,B)P(B,C)}{P(B)} = \frac{\big(P(A\mid B)P(B)\big)\big(P(C\mid B)P(B)\big)}{P(B)}$$
$$= P(A\mid B)P(B)P(C\mid B) = P(A)P(B\mid A)P(C\mid B)$$

Both sets of tables correspond to P(A,B,C)

Junction Trees and Belief Propagation

- The separator contains the information that is common to a pair of adjacent clusters
- Information propagates from cluster U to cluster V via separators and nodes along the path from U to V
- If a node A is in U and V but not in all intermediate clusters then information U has about A cannot propagate to V. Information about A is "forgotten" before V is reached.
- The junction tree property ensures that information entered at U and relevant to belief in V is not forgotten along the path from U to V

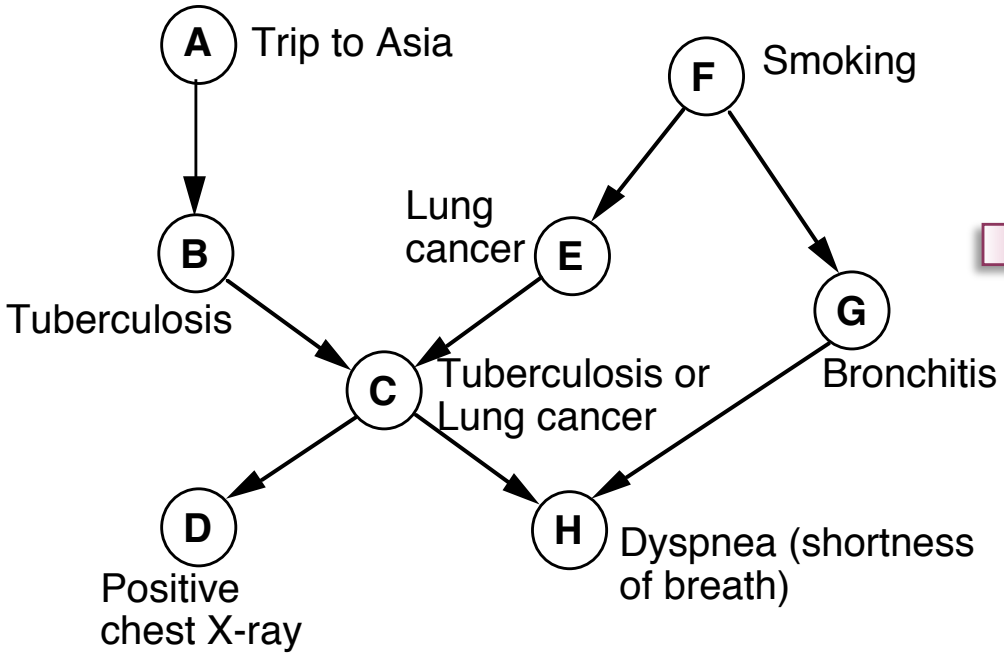
Belief Propagation Steps

- Basic steps of belief propagation for a Bayesian network B=(G,P) over universe of random variables U:
  1. Construct an initial junction tree J that corresponds to P(U)
  2. Propagate beliefs by message-passing until each cluster and separator contains the marginal distribution over its random variables
  3. Incorporate evidence on a random variable by entering a finding in a cluster containing the random variable
  4. Propagate beliefs again to update beliefs on all the cluster tables
- The key to the belief propagation algorithm is a message-passing scheme that
  - Preserves the joint probability distribution on U
  - Finishes with each cluster containing the joint distribution on random variables in the cluster given the findings

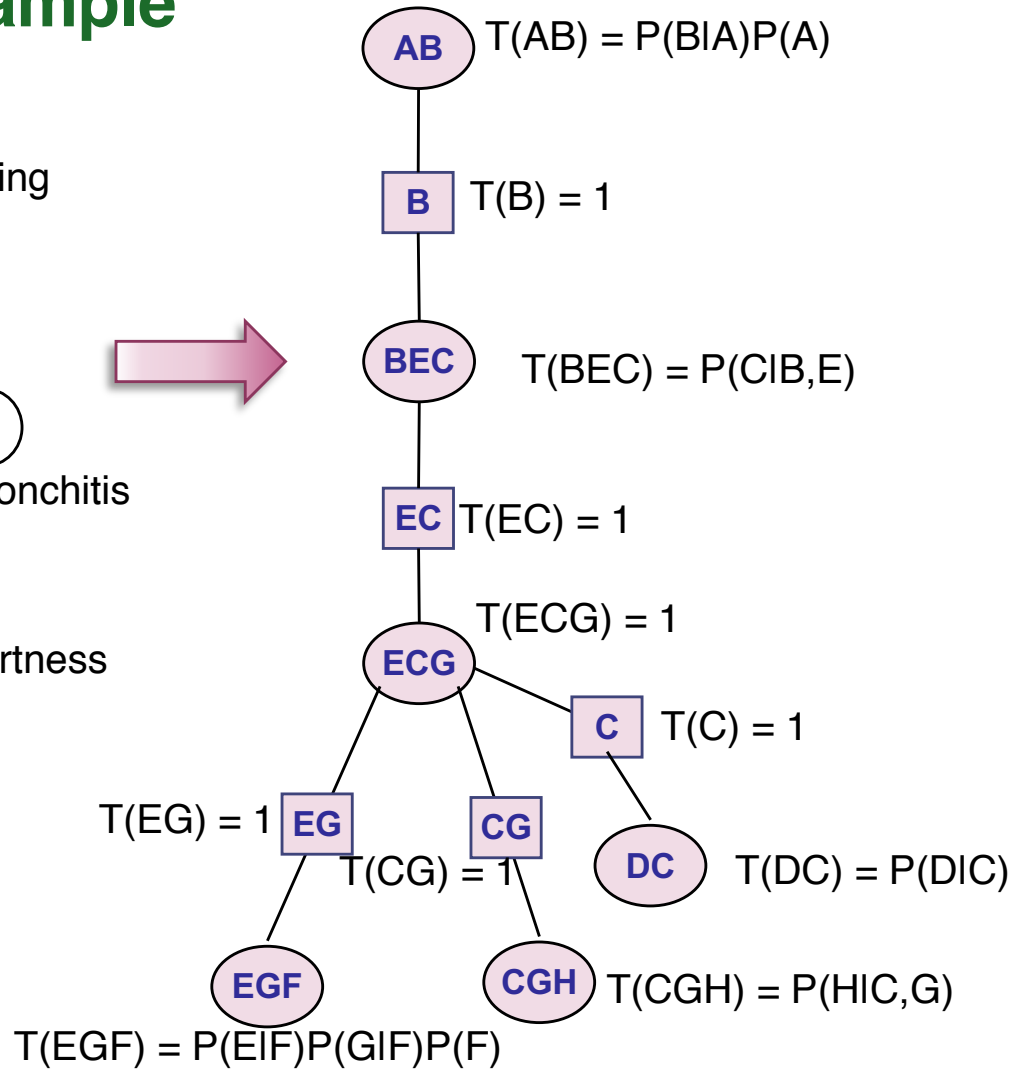
Initial Belief Assignment

- Construct a junction tree so that for each factor of the factored representation there is a cluster containing all random variables mentioned in the factor
  - In a Bayesian network for each node there must be a cluster containing that node and its parents
  - In a Markov network for each potential $\psi_C(x_C)$ there must be a cluster containing all random variables in C
- For each factor F, construct a belief table T_F
  - In a Bayesian network, for each random variable A, construct a belief table T_A over (A,pa(A)) with values P(A | pa(A))
  - In a Markov network, for each clique A, construct a belief table T_C over C $\psi_C(x_C)$
- Initial belief assignment:
  - Assign belief tables for separators with all values equal to 1.
  - Assign each table T_F constructed above to a cluster C_F in the junction tree that contains all the random variables mentioned in T_F
  - For each cluster, form a belief table that is the product of all the tables assigned to the cluster, or all 1's if no tables were assigned to the cluster
- This junction tree corresponds to the probability distribution represented by the graphical model

Example



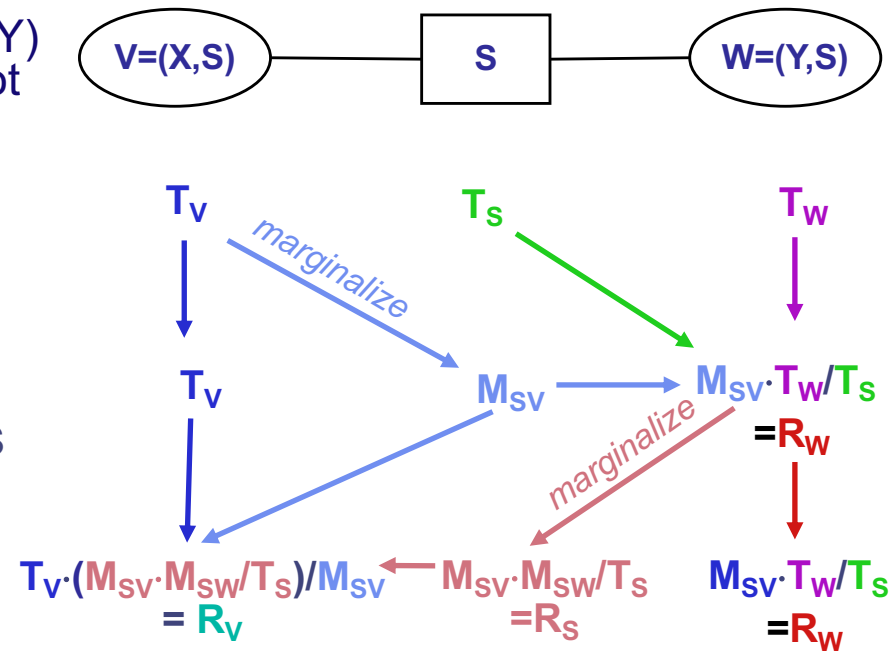
Bayesian Network



Junction Tree with Initial Beliefs

Propagation Between Two Clusters

- Begin with belief tables corresponding to P(X,S,Y) but for which cluster and separator tables are not necessarily marginal distributions over their universes
  - $P(X,S,Y) = T_V \cdot T_W / T_S$
  - T_V is not necessarily equal to P(X,S)
  - T_W is not necessarily equal to P(Y,S)
- Create new belief tables R_S, R_V and R_S:
  - M_{SV} & M_{SW} are the *marginal* tables of T_V & T_W on S
  - $R_S = M_{SV} \cdot M_{SW} / T_S$
  - $R_V = M_{SW} \cdot T_V / T_S$
  - $R_W = M_{SV} \cdot T_W / T_S$
- Facts:
  - $P(X,S,Y) \propto R_V \cdot R_W / R_S$
  - R_S is marginal table of R_V over X and R_W over Y
  - From this we can show that $R_V \propto P(V)$ and $R_W \propto P(W)$
- Propagation operations:
  - W *absorbs* from V (replace T_S by M_{SV} and T_W by R_W)
  - V *absorbs* from W
  - Mutual absorption results in belief tables R_V on V, R_W on W, and R_S on S



Absorption in a nutshell:

- *Replace separator by marginal table for cluster being absorbed*
- *Multiply table doing the absorbing by marginal table of cluster being absorbed divided by original separator table*
- *The product of cluster tables divided by separator tables remains unchanged*

Absorption and Message Passing

- D4.13: Let W and V be adjacent nodes in a cluster tree and let S be their separator. Let T_W, T_V and T_S be the associated belief tables. We say W *absorbs* from V if the following operations are performed:
  - Let M_{SV} be a belief table over S obtained by marginalizing T_V over the variables V\S not in S.
  - Assign S a new belief table M_{SV}.
  - Assign W a new belief table $T_W \cdot M_{SV}/T_S$
- D4.14: A link between W and V is *supportive* if it allows absorption in both directions. That is, the link is supportive if every zero entry in T_S corresponds to zero entries in both T_V and T_W.
- Lemma 4.2: Supportiveness is preserved under absorption
- Theorem 4.3: Let T be a supportive cluster tree. Then the product of all cluster tables divided by the product of all separator tables is invariant under absorption.
- D4.15: A cluster tree is *consistent* if T_S is the marginal table of both T_V marginalized over V\S and T_W marginalized over W\S.
- D4.16: A cluster tree is *globally consistent* if for any nodes V and W with intersection I, the marginal table of V over V\I and W over W\I are equal.
- Theorem 4.4: A consistent junction tree is globally consistent.

The Propagation Algorithm

- The algorithm - Do until done:
  - Pick any pair of neighboring clusters (V,W) such that W is eligible to absorb from V
  - V requests W to absorb from V. That is, V marginalizes over all variables not shared with W, replaces the separator table with the marginal table, and multiplies W's table by the marginal table divided by the separator table.
- Comments on the algorithm:
  - Cluster W is eligible to absorb from cluster V if:
    - » The link between W and V is supportive; and
    - » W has not already absorbed from V; and
    - » Either W is V's only neighbor or V has absorbed from all its neighbors other than W
  - Any allowable order will work.
  - The theorems imply:
    - » In a supportive cluster tree this scheme can always continue until messages have been passed in both directions on each link
    - » On termination, if no findings have been entered, the cluster tables will contain the cluster prior probability distributions for the Bayesian network
    - » On termination, if findings have been entered, the cluster tables will be proportional to the cluster conditional distributions given all findings

Example: Chest Clinic Junction Tree

1. Eligible to absorb:
   a. (BEC) from (AB)
   b. (ECG) from (EGF)
   c. (ECG) from (CGH)
   d. (ECG) from (DC)

   Choose (BEC) ← (AB)
2. Eligible to absorb:
   a. (ECG) from (EGF)
   b. (ECG) from (CGH)
   c. (ECG) from (DC)
   d. (ECG) from (BEC)

   Choose (ECG) ← (EGF)
3. Eligible to absorb:
   a. (ECG) from (CGH)
   b. (ECG) from (DC)
   c. (ECG) from (BEC)

   Choose (ECG) ← (CGH)
4. Eligible to absorb:
   a. (ECG) from (DC)
   b. (ECG) from (BEC)

   Choose (ECG) ← (DC)
5. Eligible to absorb:
   a. (ECG) from (BEC)
   b. (BEC) from (ECG)

   Choose (ECG) ← (BEC)
6. Eligible to absorb:
   a. (BEC) from (ECG)
   b. (EGF) from (ECG)
   c. (CGH) from (ECG)
   d. (DC) from (ECG)

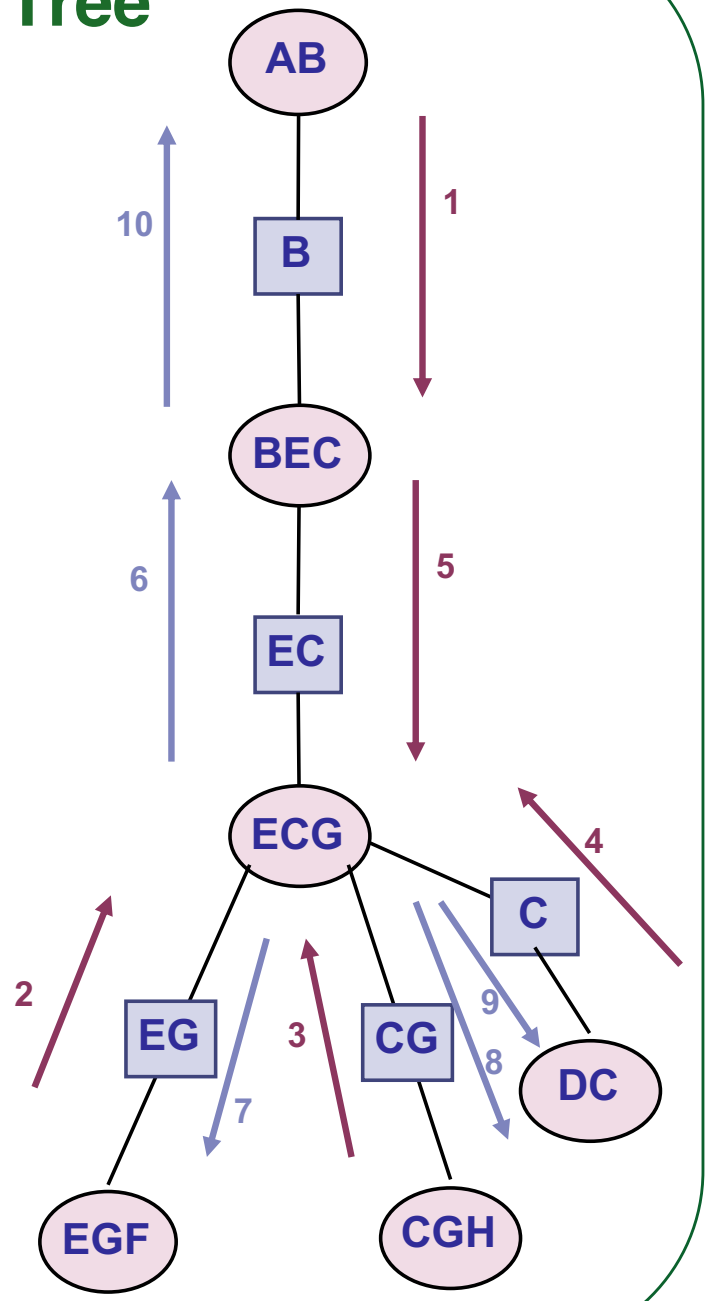
   Choose (BEC) ← (ECG)
7. Eligible to absorb:
   a. (EGF) from (ECG)
   b. (CGH) from (ECG)
   c. (DC) from (ECG)
   d. (AB) from (BEC)

   Choose (EGF) ← (ECG)
8. Eligible to absorb:
   a. (CGH) from (ECG)
   b. (DC) from (ECG)
   c. (AB) from (BEC)

   Choose (CGH) ← (ECG)
9. Eligible to absorb:
   a. (DC) from (ECG)
   b. (AB) from (BEC)

   Choose (DC) ← (ECG)
10. Eligible to absorb:
   a. (AB) from (BEC)

   Choose (AB) ← (BEC)



Steps in Junction Tree Propagation

1. Transform graphical model into junction tree
   - Permanent part of the knowledge representation
   - Changes only if the graph changes

   Coming next

2. Use local local message-passing algorithm to propagate beliefs in the junction tree
   - Each cluster and each separator has a local belief table
   - Local belief table is proportional to joint probability of nodes in cluster / separator
   - Product of cluster tables divided by product of separator tables is proportional to joint probability distribution or BN
3. Query on any node or any set of nodes in same cluster can be computed from cluster table

   We know how to do this

Making Belief Propagation More Efficient

- Lazy propagation
  - Do only the propagation steps needed to answer a specific query of interest
- Exploiting local structure
  - Context-specific independence – values in some tables are repeated. We may be able to factor repeated values out of sum-times-product formula, perform sum, and multiply after summation.
    - » This can yield computational savings when there are many repeated entries, but it may not be straightforward to identify and exploit cases where savings can be achieved.
  - Some algorithms are designed to exploit constraints (tables where many entries are zero)
    - » Multiplication: Instead of multiplying by zero, set value to zero
    - » Marginalization: Do not add zero values when marginalizing

Unit 4 Outline

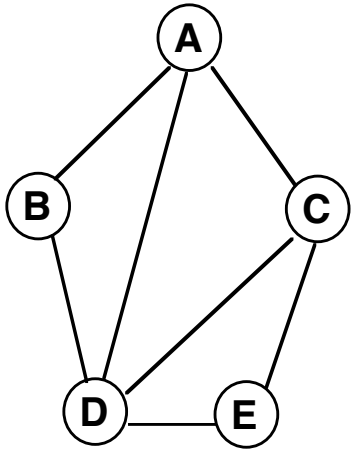
- The Inference Problem
- Pearl's Belief Propagation Algorithm
- Junction Tree Algorithm
  - Propagation in Junction Trees
  - Constructing the Junction Tree
  - Additional topics:
    - » Virtual (likelihood) evidence & soft evidence
    - » MAP / MPE

Steps in Constructing the Junction Tree

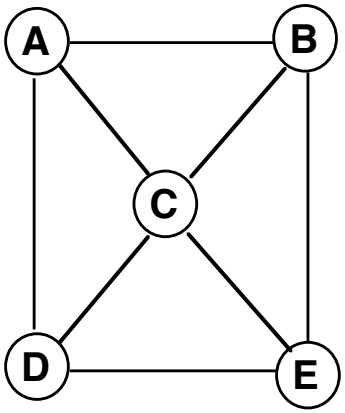
1. If the graph has directed links, turn into an undirected graph by removing arrows and moralizing (marrying coparents);
2. Triangulate the graph;
3. Find the cliques of the triangulated graph;
4. Connect the cliques to form a junction tree.

Triangulation

- D4.17: An undirected graph is triangulated if any cycle of length > 3 has a chord



Triangulated



Not triangulated

- We triangulate the graph because we can make a junction tree from a triangulated graph

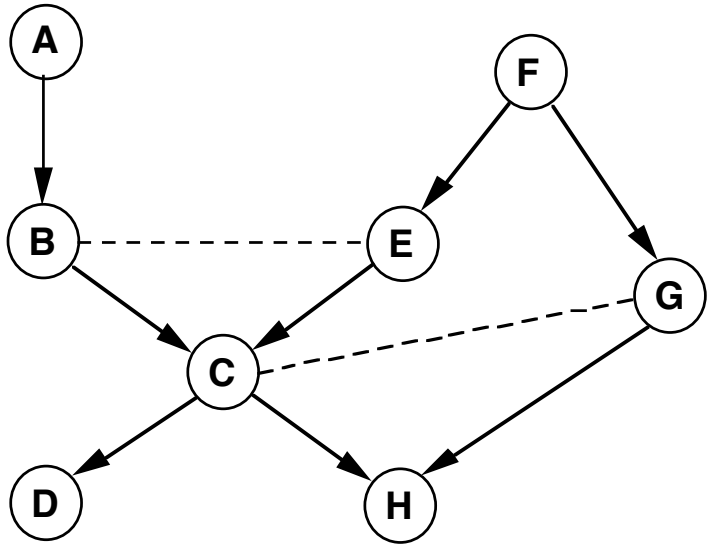
Triangulating the Graph

- The *elimination algorithm* to triangulate a graph:
  - Begin with any node A
  - Add links so that all neighbors of A are linked to each other
    - » If no links need to be added then A cannot be part of a chordless cycle of length > 3
  - Remove A together with all links to A
  - Continue (by selecting a node not already eliminated) until all nodes have been eliminated
  - The original graph together with all the links that were added during the elimination process is triangulated
- Theorem 4.5: A graph is *triangulated* if and only if all the nodes can be eliminated without adding any links
  - Whether links need to be added in elimination algorithm depends on order in which nodes are visited
- *Maximum cardinality search*. Obtain an ordering of the vertices in G=(V,E) as follows:
    - » assign 1 to an arbitrary vertex
    - » select the next vertex as the vertex adjacent to the largest number of previously numbered vertices, breaking ties arbitrarily
- Theorem 4.6: If nodes are ordered by *maximum cardinality search* and visited in reverse order, then:
  - If graph is already triangulated then no new arcs will be added
  - Node ordering can be used to construct clusters for junction tree

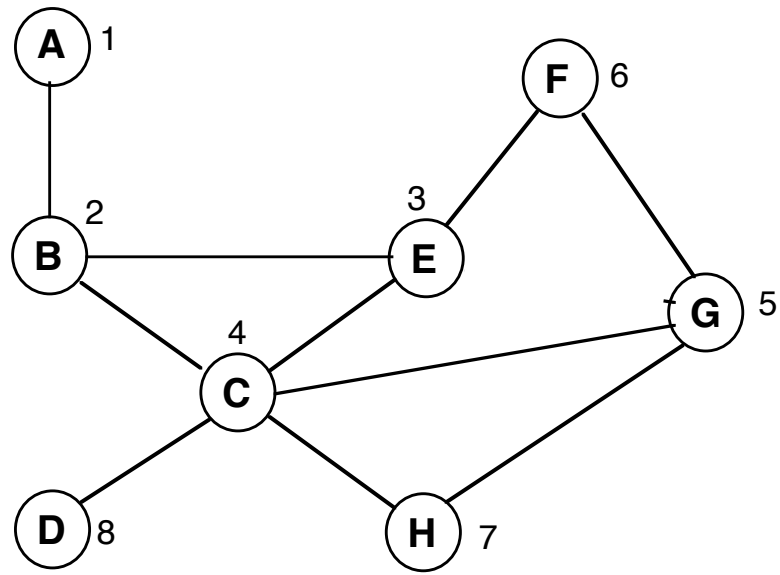
How to Construct the Junction Tree

1. "Moralize" the graph: add undirected arcs between parents of common children
   - An undirected graph representation has to encode dependency between parents of common children
   - A node's Markov blanket consists of its parents, its children, and the other parents of its children
   - A node is conditionally independent of all other nodes conditional on its Markov blanket
   - In the moral graph a node's neighbors are its Markov blanket
2. Remove all the arrowheads. The graph is now an undirected graph.
3. Order the nodes by maximum cardinality search
4. Use the elimination algorithm to triangulate the graph
5. Find the cliques of the triangulated graph.
6. Order the cliques according to their highest-numbered vertices. Go through the cliques in order, assigning a parent to each clique among the previously numbered cliques. A clique's intersection with its parent must contain its intersection with every previously numbered clique. (This makes the resulting tree have the junction tree property.)
7. Arrange as a junction tree.

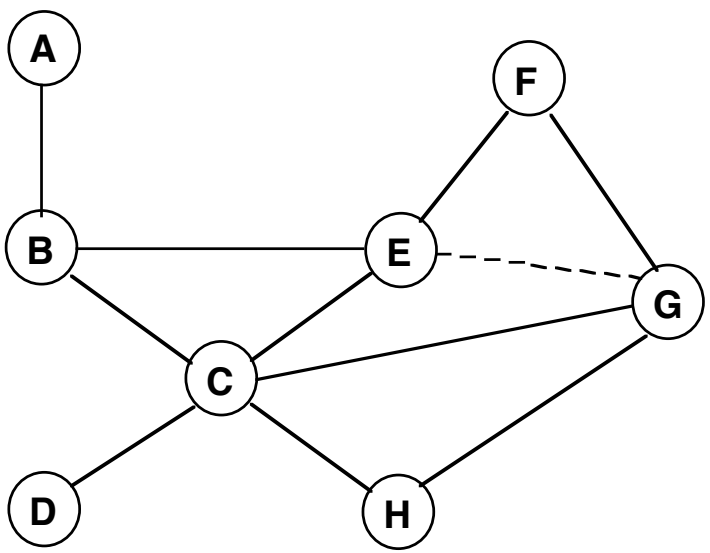
Example



Moralizing the Graph

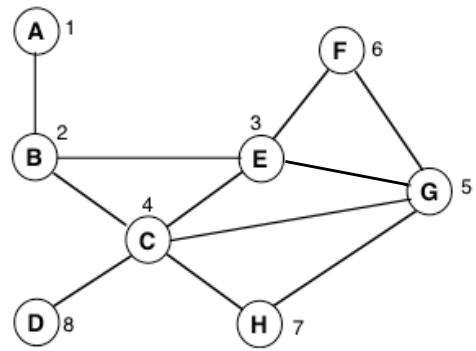


Ordering by Maximum Cardinality Search

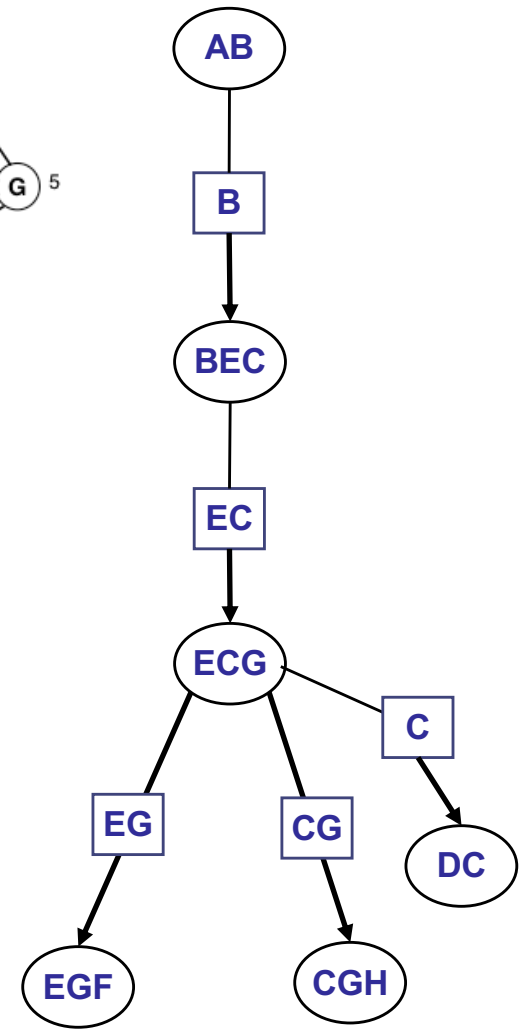


The Triangulated Graph

Example continued



Clique	Highest vertex
AB	2
BEC	4
ECG	5
EGF	6
CGH	7
DC	8



A Junction Tree

Are there alternate junction trees that could be constructed from these cliques?

Review of Junction Tree Propagation

- Construct junction tree
  - Moralize graph and remove arrows*
  - Triangulate
  - Find cliques
  - Arrange as junction tree
- Assign initial belief tables
- For each query:
  - Declare evidence arriving since last query
  - Remove evidence retracted since last query
  - Propagate
- Compute marginal probability distribution for X_t given X_e

* If there are directed edges

Graphical Probability Models in R

- Here is a link to R packages related to graphical probability models: https://cran.r-project.org/web/views/gR.html
- R package gRain (for Graphical Independence Networks) contains functions for:
  - Defining a Bayesian network (nodes, arcs, states, local distributions)
  - Reading a Bayesian network in .net format
  - Setting evidence
  - Querying for distributions of a node or set of nodes
- gRain package can be downloaded from:
  - https://cran.r-project.org/web/packages/gRain/index.html
- Journal of Statistical Software article on gRain (2009):
  - https://www.jstatsoft.org/article/view/v046i10

Complexity and Treewidth

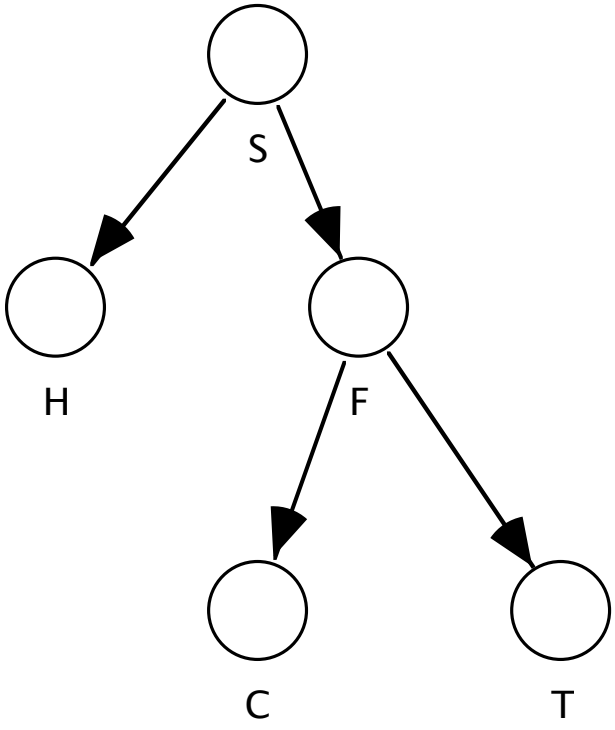
- D4.18: The *treewidth* of a triangulated graph is one less than the size of the largest clique. The treewidth of a general graph is the minimum treewidth of all triangulated supergraphs.
- Complexity of junction tree algorithm
  - For models in which the treewidth is less than a fixed bound, junction tree inference has complexity polynomial in the size of the graph
  - Junction tree inference is exponential in the treewidth of the graph
- "There is no class of graphical models consisting of binary variables with unbounded treewidth in which inference can be performed in time polynomial in the treewidth." (Chandrasekeran et al., 2008)
  - Note that this result applies to any exact inference algorithm, not just to junction tree inference.
- Bottom line: efficient exact inference algorithms exist only for classes of graphical models with bounded treewidth

Unit 4 Outline

- The Inference Problem
- Pearl's Belief Propagation Algorithm
- Junction Tree Algorithm
  - Propagation in Junction Trees
  - Constructing the Junction Tree
  - Additional topics:
    - » Virtual (likelihood) evidence & soft evidence
    - » MAP / MPE

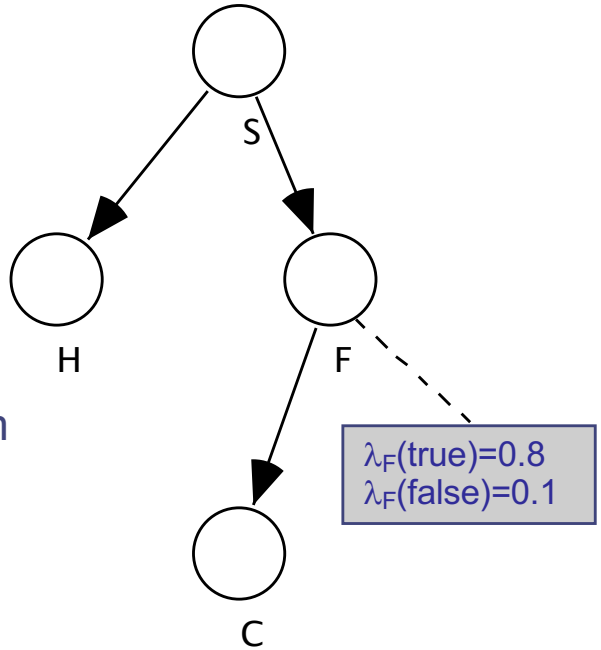
Virtual Evidence

- Ten year old Leslie comes downstairs complaining of a headache and chills. You put your hand on her forehead and decide that she probably has a fever. What's the probability she's sick enough to stay home from school?
- We draw a Bayesian network with nodes S (sick?), H (headache?), F (fever?), C (chills?). How do we process the evidence that her forehead "feels hot?"
- We could add a "virtual" child of F called T (touch forehead). We would assign possible values (e.g., very hot, hot, warm, cool) for T and assess P(T | F) for each value of F and T
- But these categorizations are arbitrary. Is her forehead hot, very hot, or warm? Why should we have to assess P(F|T) for values of T that did not occur?



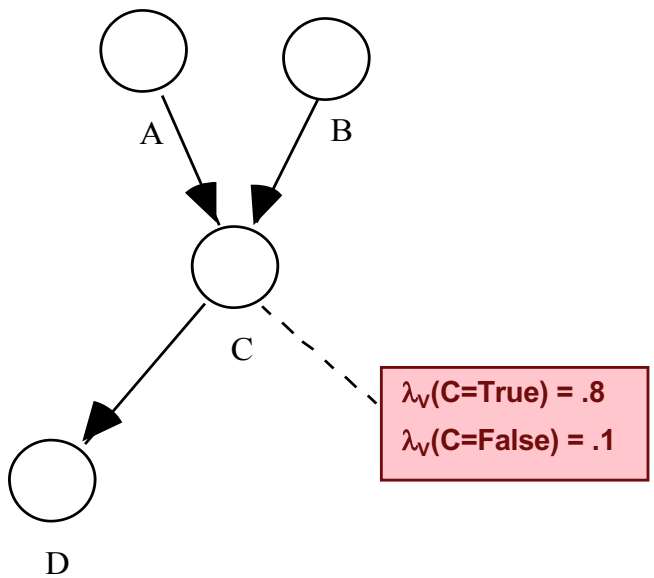
More on Virtual Evidence

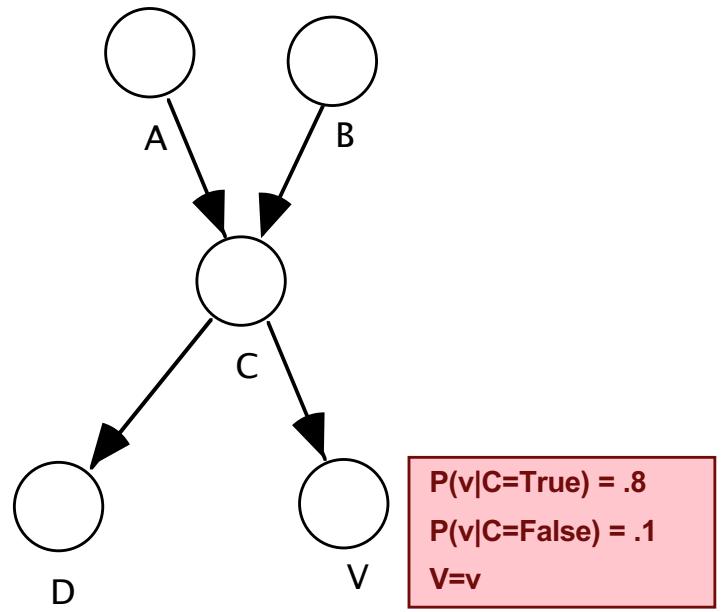
- A more natural model for inconclusive evidence is to leave off the T variable and introduce *virtual (or likelihood) evidence* for F.
- To declare virtual evidence, we assess the relative probabilities of the her forehead feeling the way it does given fever or no fever.
- Some facts about virtual evidence λ values:
  - Only relative values matter
    - » (.8, .1), (1.6, .2) and (.4, .05) all give the same result when plugged into the propagation algorithm
  - To assess virtual evidence we need to consider only the way her forehead actually feels, not all the ways it might have felt
  - Virtual evidence at a node does not deactivate diverging or serial chains through the node
  - Virtual evidence below a node activates converging chains through that node
- Many BN software packages support virtual evidence



Likelihood Evidence and Virtual Nodes

- The following two models are equivalent.

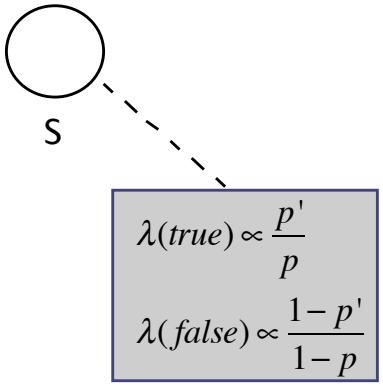




- Virtual evidence activates head-to-head links and does not block other links
  - Chain A → C → B is activated by V
  - Chain A → C → D is not blocked by V

Soft Evidence

- Soft evidence (aka Jeffrey's evidence) is evidence specifying a new probability distribution for a random variable
  - E.g., we want to declare soft evidence to change the probability Leslie is sick enough to stay home from school from *p* to *p*'
- Soft evidence can be implemented via virtual evidence
  - Set likelihood ratio to produce the intended probability



(Valtorta, 2002)

Implementing Virtual Evidence with Junction Tree

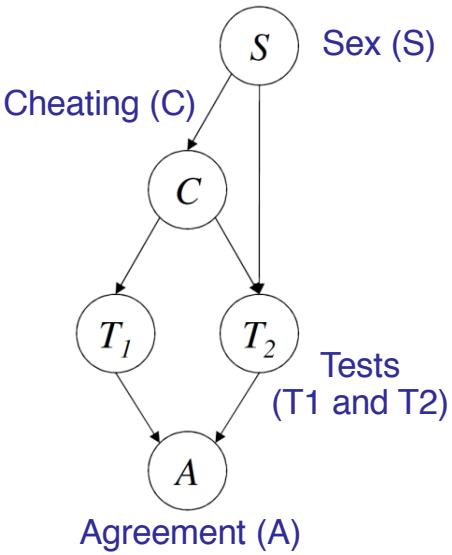
- Virtual evidence on random variable X consists of a vector of likelihoods $\lambda_X(X=x)$, one for each state x of X
- To implement virtual evidence with the junction tree algorithm:
  - Choose a cluster that W = (X, Y) that contains X
  - Form a "virtual finding" table $F_W(x,y) = \lambda_X(X=x)$
    - » Set every entry corresponding to X=x to the value $\lambda_X(X=x)$
  - Replace the existing table T_W by $T_W F_W$
  - Propagate using the junction tree algorithm
- Note that a likelihood finding is entered in only one cluster containing the variable
- To implement soft evidence to set the distribution of X to Q(X), calculate the marginal distribution P(X) and enter a virtual finding $\lambda_X(X=x) = Q(x)/P(x)$

MPE and MAP

- Belief propagation algorithms can be modified to handle MAP / MPE
- *MAP and MPE are different problems!*
  - **Most probable explanation (MPE)**: Given evidence $X_e=x_e$, find the most probable instantiation of the non-evidence random variables (X_t,X_o)
  - **Maximum a posteriori (MAP)**: Given evidence $X_e=x_e$, find the most probable instantiation of a target random variable X_t

MPE ≠ MAP
Example: Detecting Cheating

This example is due to Vibhav Gogate



S	C	T_2	$\theta_{t_2\|c,s}$
male	yes	+ve	.80
male	yes	–ve	.20
male	no	+ve	.20
male	no	–ve	.80
female	yes	+ve	.95
female	yes	–ve	.05
female	no	+ve	.05
female	no	–ve	.95

T_1	T_2	A	$\theta_{a\|t_1,t_2}$
+ve	+ve	yes	1
+ve	+ve	no	0
+ve	–ve	yes	0
+ve	–ve	no	1
–ve	+ve	yes	0
–ve	+ve	no	1
–ve	–ve	yes	1
–ve	–ve	no	0

S	θ_s
male	.55
female	.45

S	C	$\theta_{c\|s}$
male	yes	.05
male	no	.95
female	yes	.01
female	no	.99

C	T_1	$\theta_{t_1\|c}$
yes	+ve	.80
yes	–ve	.20
no	+ve	.20
no	–ve	.80

- Person takes test and administrator says two tests agree (A=yes)
  - MPE: S=female, C=no, T1=negative, T2=negative
  - MAP for sex and cheating: S=male, C=no

Cheating Example MAP and MPE - Details

S	C	T1	T2	A	P(S)	P(C|S)	P(T1|C)	P(T2|S,C)	P(A|T1,T2)	P(S,C,T1,T	P(S,C,T1,T2 |A=yes)	P(S,C |A=yes)
male	yes	+ve	+ve	yes	0.55	0.05	0.80	0.80	1	0.0176	0.0244	0.0260
female	yes	+ve	+ve	yes	0.45	0.01	0.80	0.95	1	0.0034	0.0047	0.0048
male	no	+ve	+ve	yes	0.55	0.95	0.20	0.20	1	0.0209	0.0290	**0.4931**
female	no	+ve	+ve	yes	0.45	0.99	0.20	0.05	1	0.0045	0.0062	0.4761
male	yes	-ve	+ve	yes	0.55	0.05	0.20	0.80	0	0	0	0.0260
female	yes	-ve	+ve	yes	0.45	0.01	0.20	0.95	0	0	0	0.0048
male	no	-ve	+ve	yes	0.55	0.95	0.80	0.20	0	0	0	0.4931
female	no	-ve	+ve	yes	0.45	0.99	0.80	0.05	0	0	0	0.4761
male	yes	+ve	-ve	yes	0.55	0.05	0.80	0.20	0	0	0	0.0260
female	yes	+ve	-ve	yes	0.45	0.01	0.80	0.05	0	0	0	0.0048
male	no	+ve	-ve	yes	0.55	0.95	0.20	0.80	0	0	0	0.4931
female	no	+ve	-ve	yes	0.45	0.99	0.20	0.95	0	0	0	0.4761
male	yes	-ve	-ve	yes	0.55	0.05	0.20	0.20	1	0.0011	0.0015	0.0260
female	yes	-ve	-ve	yes	0.45	0.01	0.20	0.05	1	4.5E-05	0.0001	0.0048
male	no	-ve	-ve	yes	0.55	0.95	0.80	0.80	1	0.3344	0.4641	0.4931
female	no	-ve	-ve	yes	0.45	0.99	0.80	0.95	1	0.3386	**0.4699**	0.4761
male	yes	+ve	+ve	no	0.55	0.05	0.80	0.80	0	0	0	0
female	yes	+ve	+ve	no	0.45	0.01	0.80	0.95	0	0	0	0
male	no	+ve	+ve	no	0.55	0.95	0.20	0.20	0	0	0	0
female	no	+ve	+ve	no	0.45	0.99	0.20	0.05	0	0	0	0
male	yes	-ve	+ve	no	0.55	0.05	0.20	0.80	1	0.0044	0	0
female	yes	-ve	+ve	no	0.45	0.01	0.20	0.95	1	0.0009	0	0
male	no	-ve	+ve	no	0.55	0.95	0.80	0.20	1	0.0836	0	0
female	no	-ve	+ve	no	0.45	0.99	0.80	0.05	1	0.0178	0	0
male	yes	+ve	-ve	no	0.55	0.05	0.80	0.20	1	0.0044	0	0
female	yes	+ve	-ve	no	0.45	0.01	0.80	0.05	1	0.0002	0	0
male	no	+ve	-ve	no	0.55	0.95	0.20	0.80	1	0.0836	0	0
female	no	+ve	-ve	no	0.45	0.99	0.20	0.95	1	0.0846	0	0
male	yes	-ve	-ve	no	0.55	0.05	0.20	0.20	0	0	0	0
female	yes	-ve	-ve	no	0.45	0.01	0.20	0.05	0	0	0	0
male	no	-ve	-ve	no	0.55	0.95	0.80	0.80	0	0	0	0
female	no	-ve	-ve	no	0.45	0.99	0.80	0.95	0	0	0	0

P(S,C!A=yes)

MAP for (S,C) occurs at S=male, C=no

MPE occurs at S=female, C=no, T1=negative, T2=negative

Most Probable Explanation (MPE)

- The junction tree algorithm becomes a MPE algorithm if we replace marginalization by maximization in the propagation algorithm
- This algorithm is known as “max propagation” and is closely related to the “Viterbi algorithm” from dynamic programming
- The MPE algorithm finds the most probable instantiation for each random variable given the evidence

Maximum a Posteriori (MAP)

- We can find MAP using the junction tree algorithm if MAP variables are all in the same clique
- We can also find MAP by:
  - marginalizing out all non-MAP variables
  - performing MPE on reduced junction tree

Summary and Synthesis

- The belief propagation problem
  - Find marginal distribution on target variable(s) X_t given values of or virtual evidence about evidence variables X_e
  - Find highest probability instantiation of all other variables given values of evidence variables X_e (or virtual evidence)
- Belief propagation can be done with local message passing algorithms in singly connected networks
  - Only one chain by which evidence can travel between two nodes
- Junction tree algorithm
  - Turns multiply connected graph into singly connected junction tree
  - Nodes in junction tree are clusters of variables
  - Beliefs are propagated using local message passing algorithm on tree of cliques

References for Unit 4

- Jensen and Nielsen text (1st and 2nd editions)
- Neapolitan, R. Probabilistic Reasoning in Expert Systems: Theory and Algorithms. Wiley, 1990. Covers Pearl and clique tree algorithms in thorough detail.
- Neapolitan, R. Learning Bayesian Networks. Prentice Hall, 2004.
- Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988. The first book popularizing Bayesian networks; presents Pearl's algorithm for singly connected Bayesian networks.
- Valtorta, M., Kim, Y.-G., & Vomlel, J., 2002. Soft evidential update for probabilistic multiagent systems. *International Journal of Approximate Reasoning*, *29*(1), 71–106. https://doi.org/10.1016/S0888-613X(01)00056-1
- V. Chandrasekaran, N. Srebro, and P. Harsha. 2008. Complexity of Inference in Graphical Models. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)* http://users.cms.caltech.edu/~venkatc/csh_compinf_uai08.pdf