Probabilistic Graphical Models

Raquel Urtasun and Tamir Hazan

TTI Chicago

April 25, 2011

Summary

This week we saw:

- VE via message-passing.
- Sum-product BP.

Today we are going to see:

- Belief update algorithm.
- How to do more complex inference tasks.
- How to compute the clique tree.
- Junction tree algorithm

Previous algorithm: Sum product BP

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

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

1 Initialize-Cliques
2 **while** exist i, j such that i is ready to transmit to j
3 $\delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow$ SP-Message(i, j)
4 **for** each clique i
5 $\beta_i \leftarrow \psi_i \cdot \prod_{k \in \text{Nb}_i} \delta_{k \to i}$
6 **return** $\{\beta_i\}$

(calibration algorithm)

Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)

1 $\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
2 $\tau(\boldsymbol{S}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi(\boldsymbol{C}_i)$
3 **return** $\tau(\boldsymbol{S}_{i,j})$

(message computation)

Clique Tree measure

Def: We define the **clique tree measure** of a calibrated tree $\mathcal{T}$ as

$$\beta_{\mathcal{T}} = \frac{\prod_{i \in \mathcal{V}_{\mathcal{T}}} \beta_i(\mathbf{C}_i)}{\prod_{(i-j) \in \mathcal{E}_{\mathcal{T}}} \mu_{i,j}(\mathbf{S}_{i,j})}$$

where

$$\mu_{i,j}(\mathbf{S}_{i,j}) = \sum_{\mathbf{C}_i - \mathbf{S}_{i,j}} \beta_i(\mathbf{C}_i) = \sum_{\mathbf{C}_i - \mathbf{S}_{i,j}} \beta_j(\mathbf{C}_j)$$

Theorem: Let $\mathcal{T}$ be a clique tree over Φ, and let $\beta_i(\mathbf{C}_i)$ be a set of calibrated potentials for $\mathcal{T}$. Then $\hat{P}_\Phi(\mathcal{X}) \propto \beta_{\mathcal{T}}$ iff for each $i \in \mathcal{V}_{\mathcal{T}}$ we have that $\beta_i(\mathbf{C}_i) \propto \hat{P}_\Phi(\mathbf{C}_i)$.

For a proof see page 333 in D. Koller book.

We can view the clique tree as an alternative representation of the joint measure, which directly gives the clique marginals.

Belief update

- Derive an alternative message passing which is mathematically equivalent to the previous one.
- Consider again the previous algorithm: two messages are passed along $(i - j)$.
- Assume that the first message is pass from $\mathbf{C}_j$ to $\mathbf{C}_i$.
- The return message will be passed when $\mathbf{C}_i$ has received the messages from all its neighbors.
- At this point $\mathbf{C}_i$ has all the information

$$\beta_i = \psi_i \prod_{k \in Nb_i} \delta_{k \to i}$$

- To avoid double counting, the final potential β_i is not used when computing the message passed to $\mathbf{C}_j$. Thus

$$\delta_{i \to j} = \sum_{\mathbf{C}_i - \mathbf{S}_{i,j}} \psi_i \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i}$$

Message passing with division

- An alternative possibility is to multiply all the messages, and then divide by the $\delta_{j \to i}$ to avoid the double counting.
- **Def:** Let $\mathbf{X}$ and $\mathbf{Y}$ be disjoint sets of variables, and let $\phi_1(\mathbf{X}, \mathbf{Y})$ and $\phi_2(\mathbf{Y})$ be two factors. We define the factor division ϕ_2/ϕ_1 to be the factor with scope $\mathbf{X}, \mathbf{Y}$ defined as

$$\psi(\mathbf{X}, \mathbf{Y}) = \frac{\phi_1(\mathbf{X}, \mathbf{Y})}{\phi_2(\mathbf{Y})}$$

- The factor division is done component by component, and we define $0/0 = 0$.
- It is not defined if the numerator is not 0 and the denominator is.
- We can compute the message by

$$\delta_{i \to j}(\mathbf{S}_{i,j}) = \frac{\sum_{\mathbf{C}_i - \mathbf{S}_{i,j}} \beta_i}{\delta_{j \to i}(\mathbf{S}_{i,j})}$$

Example

- Chain network $A - B - C - D$.
- First messages from $\mathbf{C}_1$ to $\mathbf{C}_2$ and $\mathbf{C}_2$ to $\mathbf{C}_3$.
- What are $\delta_{1\to 2}$ and $\delta_{2\to 3}$?
- Using the previous CTree-Sum-Product-Calibrate, what's $\delta_{2\to 1}$?
- What would it be in the new sum product division algorithm?

Sum product divide message-passing

- Each clique $\mathbf{C}_i$ maintains its fully updated **beliefs** β_i.
- These beliefs are the product of its initial potential with all of the message updates it has received from its neighbors.
- Store at each sepset $\mathbf{S}_{i,j}$ the previous message $\mu_{i,j}$ that was passed along the edge $(i - j)$ regardless of the direction.
- Whenever a message is passed along the edge it's divided by the old one to avoid double counting.
- This is correct regardless of the clique that send the message along the edge.
- This algorithm can be view as maintaining a set of beliefs over the cliques in the tree.
- The message passing takes the beliefs of one clique and uses it to update the beliefs of a neighbor.
- This algorithm is called **belief update message passing**

Sum Product Divide

Algorithm 10.3 Calibration using belief propagation in clique tree

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

1 Initialize-CTree
2 **while** exists an uninformed clique in $\mathcal{T}$
3 Select $(i–j) \in \mathcal{E}_{\mathcal{T}}$
4 Send-BU-message(i, j)
5 **return** $\{\beta_i\}$

Procedure Initialize-CTree (
)

1 **for** each clique $\boldsymbol{C}_i$
2 $\beta_i \leftarrow \prod_{\phi \,:\, \alpha(\phi)=i} \phi$
3 **for** each edge $(i–j) \in \mathcal{E}_{\mathcal{T}}$
4 $\mu_{i,j} \leftarrow \mathbf{1}$

Procedure Send-BU-Msg (
  i, // sending clique
  j // receiving clique
)

1 $\sigma_{i \to j} \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i$
2 // marginalize the clique over the sepset
3 $\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i \to j}}{\mu_{i,j}}$
4 $\mu_{i,j} \leftarrow \sigma_{i \to j}$

Properties of Sum Product Divide

- This process is correct even if we send multiple messages.
- We will like to do this with a minimum number of messages.
- In the sum product BP algorithm, we said that we need to send 2 messages for each edge.
- In this case is the same. We guarantee convergence when we have a calibrated tree

$$\sum_{\mathbf{C}_i - \mathbf{S}_{i,j}} \beta_i = \mu_{i,j}(\mathbf{S}_{i,j}) = \sum_{\mathbf{C}_j - \mathbf{S}_{i,j}} \beta_j$$

- At convergence, each pair of neighboring cliques must agree on the variables in the sepset, and the message $\mu_{i,j}$ is the sepset marginal.

Clique Tree Invariant

- The Sum Product Belief algorithm can also be interpreted as a reparameterization of the distribution $\hat{P}_{\Phi}$.
- At the convergence of any belief update calibration process we have

$$\hat{P}_{\Phi}(\mathcal{X}) = \frac{\prod_{i \in \mathcal{V}_{\mathcal{T}}} \beta_i(\mathbf{C}_i)}{\prod_{(i-j) \in \mathcal{E}_{\mathcal{T}}} \mu_{i,j}(\mathbf{S}_{i,j})}$$

- The same property holds at the start of the algorithm where the numerator is

$$\prod_{i \in \mathcal{V}_{\mathcal{T}}} \psi_i = \prod_{i \in \mathcal{V}_{\mathcal{T}}} \prod_{\phi : \alpha(\phi) = i} \phi = \prod_{\phi \in \mathbf{\Phi}} \phi = \hat{P}_{\Phi}$$

  and the denominator is 1.
- This property is called the **Clique tree invariant**, and is preserved across the entire belief update message passing process.

Clique Tree Invariant more formally

Theorem: Let $\mathcal{T}$ be a clique tree, and $\{\beta_i\}, \{\mu_{i,j}\}$ be some set of clique and sepset beliefs for the tree such that the following eq. holds.

$$\hat{P}_\Phi(\mathcal{X}) = \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i(\mathbf{C}_i)}{\prod_{(i-j) \in \mathcal{E}_\mathcal{T}} \mu_{i,j}(\mathbf{S}_{i,j})}$$

Assume that we now execute a BP message passing step from $\mathbf{C}_i$ to $\mathbf{C}_j$, and let $\{\beta'_j\}, \{\mu'_{i,j}\}$ be the new beliefs after that step. Then the same equation holds for $\{\beta'_j\}, \{\mu'_{i,j}\}$.

Proof: A message from $\mathbf{C}_i$ to $\mathbf{C}_j$ only changes β_j and $\mu_{i,j}$. By definition of the message send

$$\beta'_j = \beta_j \frac{\mu'_{i,j}}{\mu_{i,j}}$$

Thus

$$\frac{\beta_j}{\mu_{i,j}} = \frac{\beta'_j}{\mu'_{i,j}}$$

Therefore the clique tree invariant property holds initially and for every message-passing step.

Building equivalence Sum product and Belief update

- Both sum-product and belief update should be equal to the clique marginals.
- We have

$$\beta_i = \psi_i \cdot \prod_{(i-j)\in\mathcal{E}_\mathcal{T}} \delta_{j\to i}$$

substituting this into the clique tree invariant we have

$$\hat{P}_\Phi(\mathcal{X}) = \frac{\prod_{i\in\mathcal{V}_\mathcal{T}} \beta_i}{\prod_{(i-j)\in\mathcal{E}_\mathcal{T}} \mu_{i,j}} = \frac{\prod_{\mathbf{C}_i} \psi_i \cdot \left(\prod_{j:(i-j)\in\mathcal{E}_\mathcal{T}} \delta_{j\to i}\right)}{\prod_{(i-j)\in\mathcal{E}_\mathcal{T}} \mu_{i,j}} = \frac{\prod_{\mathbf{C}_i} \psi_i \cdot \left(\prod_{(i-j)\in\mathcal{E}_\mathcal{T}} \delta_{j\to i} \cdot \delta_{i\to j}\right)}{\prod_{(i-j)\in\mathcal{E}_\mathcal{T}} \mu_{i,j}}$$

- On the other hand

$$\hat{P}_\Phi(\mathcal{X}) = \prod_{i\in\mathcal{V}_\mathcal{T}} \psi_i$$

- It follows that

$$\prod_{(i-j)\in\mathcal{E}_\mathcal{T}} \delta_{j\to i}(\mathbf{S}_{i,j}) \cdot \delta_{i\to j}(\mathbf{S}_{i,j}) = \prod_{(i-j)\in\mathcal{E}_\mathcal{T}} \mu_{i,j}(\mathbf{S}_{i,j})$$

- As the messages have different scopes, then $\delta_{j\to i}(\mathbf{S}_{i,j}) \cdot \delta_{i\to j}(\mathbf{S}_{i,j}) = \mu_{i,j}(\mathbf{S}_{i,j})$.

Equivalence of Sum Product and Belief-update messages

Theorem: Consider a set of sum-prod init. potentials $\{\psi_i : i \in \mathcal{V}_\mathcal{T}\}$ and messages $\{\delta_{i\to j}, \delta_{j\to i} : (i-j) \in \mathcal{E}_\mathcal{T}\}$ and a set of beliefs $\{\beta_i : i \in \mathcal{V}_\mathcal{T}\}$ and messages $\{\mu_{i,j} : (i-j) \in \mathcal{E}_\mathcal{T}\}$ for which the following equations hold

$$\beta_i = \psi_i \cdot \prod_{(i-j)\in\mathcal{E}_\mathcal{T}} \delta_{j\to i}$$

$$\mu_{i,j}(\mathbf{S}_{i,j}) = \delta_{j\to i}(\mathbf{S}_{i,j}) \cdot \delta_{i\to j}(\mathbf{S}_{i,j})$$

For any pair of neighboring cliques $\mathbf{C}_i, \mathbf{C}_j$, let $\{\delta'_{i\to j}, \delta'_{j\to i} : (i-j) \in \mathcal{V}_\mathcal{T}\}$ be the set of sum product messages from a single run of the message, and let $\{\beta'_i : i \in \mathcal{V}_\mathcal{T}\}$ and messages $\{\mu'_{i,j} : (i-j) \in \mathcal{E}_\mathcal{T}\}$ be the set of beliefs updated with a single run as well. Then the two equations also hold for the new beliefs $\delta'_{i\to j}, \beta'_i, \mu'_{i,j}$.

- There is an equivalence between both algorithms, and thus they both converge to the marginals.
- We can derive a two pass algorithm.

Two pass algorithm

Algorithm 10.4 Upward-downward belief propagation algorithm in clique tree

Procedure CTree-BU-two-pass (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)
1 Initialize-Cliques
2 Select some clique $\boldsymbol{C}_r$ to be the root clique
3 // Upward pass
4 **repeat**
5 Select $i \in \mathcal{V}_{\mathcal{T}}$ that has received all messages from $\{\boldsymbol{C}_j : j \in \text{Nb}_i - p_r(i)$
6 Send-BU-message$(i, p_r(i))$
7 **while** exist i, j such that i is ready to transmit to j
8 // Downward pass
9 Mark $\boldsymbol{C}_r$ as informed
10 **while** exists uninformed $\boldsymbol{C}_i$ such that $\boldsymbol{C}_{p_r(i)}$ is informed
11 Send-BU-message$(p_r(i), i)$
12 **return** $\{\beta_i\}$

Procedure Send-BU-Msg (
  i, // sending clique
  j // receiving clique
)
1 $\sigma_{i \to j} \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i$
2 // marginalize the clique over the sepset
3 $\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i \to j}}{\mu_{i,j}}$
4 $\mu_{i,j} \leftarrow \sigma_{i \to j}$

Answering queries: new evidence

- All variables that we were interested to do a query over were in the same clique.
- Suppose that we have done our calibration and now we receive more evidence, how can we modify the calibrated tree?
- The most naive approach is run again the sum-product BP or belief update algorithm.
- We can obtained $\hat{P}_\Phi(\mathcal{X}, Z = z)$ by zeroing out entries that are inconsistent with $Z = z$.

$$\hat{P}_\Phi(\mathcal{X}, Z = z) = \mathbb{1}\{Z = z\} \cdot \prod_{\phi \in \Phi} \phi = \mathbb{1}\{Z = z\} \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i(\mathbf{C}_i)}{\prod_{(i-j) \in \mathcal{E}_\mathcal{T}} \mu_{i,j}(\mathbf{S}_{i,j})}$$

- If calibrated before, for cliques that contain this variable, we can just multiply by $\mathbb{1}\{Z = z\}$.
- To calibrate the other cliques, we need to transmit the information from $\mathbf{C}_i$ to $\mathbf{C}_j$ via the cliques in the path.
- Thus the entire clique can be calibrated using an entire pass.

Answering queries: queries outside the clique

- If we retract evidence is not that simple, as we have lost information once we zero out.
- Consider a query $P(\mathbf{Y}|\mathbf{e})$ where $\mathbf{Y}$ is not present together in a single clique.
- Naive approach is construct a clique tree where we force all the variables to be in the same clique.
- Alternative approach is to do VE over a calibrated tree.
- Compute the joint $\hat{P}_\Phi(\mathbf{Y})$ by using the beliefs in a calibrated tree to define factors corresponding to conditional probabilities.
- Then perform variable elimination over the resulting factors.
- Show example of a chain in the board.

Out of clique algorithm

Algorithm 10.5 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
)

1 Let $\mathcal{T}'$ be a subtree of $\mathcal{T}$ such that $\boldsymbol{Y} \subseteq Scope[\mathcal{T}']$
2 Select a clique $r \in \mathcal{V}_{\mathcal{T}'}$ to be the root
3 $\Phi \leftarrow \beta_r$
4 **for** each $i \in \mathcal{V}'_{\mathcal{T}}$
5 $\phi \leftarrow \frac{\beta_i}{\mu_{i,p_r(i)}}$
6 $\Phi \leftarrow \Phi \cup \{\phi\}$
7 $\boldsymbol{Z} \leftarrow Scope[\mathcal{T}'] - \boldsymbol{Y}$
8 Let $\prec$ be some ordering over $\boldsymbol{Z}$
9 **return** Sum-Product-Variable-Elimination$(\Phi, \boldsymbol{Z}, \prec)$

Answering queries: multiple queries

- Calibrated tree $\mathcal{T}$ over Φ, and we want $\hat{P}_\Phi(X, Y)$ for every pair of variables.
- Naive: Run VE in the calib. graph for every pair of variables, i.e., $\binom{n}{2}$.
- Better approach is to execute this process gradually, constructing a table for each $\mathbf{C}_i, \mathbf{C}_j$ that contains $\hat{P}_\Phi(\mathbf{C}_i, \mathbf{C}_j)$ in the order of the distance between $\mathbf{C}_i$ and $\mathbf{C}_j$ in the tree.
- When they are neighbors $\hat{P}_\Phi(\mathbf{C}_i|\mathbf{C}_j) = \frac{\beta_j(\mathbf{C}_j)}{\mu_{i,j}(\mathbf{C}_i \cap \mathbf{C}_j)}$
- From these we can compute $\hat{P}_\Phi(\mathbf{C}_i, \mathbf{C}_j)$.
- For non neighbors, let $\mathbf{C}_l$ be the neighbor of $\mathbf{C}_j$ that is one step closer in the tree to $\mathbf{C}_i$.
- By construction we have already computed $\hat{P}_\Phi(\mathbf{C}_i, \mathbf{C}_l)$ and $\hat{P}_\Phi(\mathbf{C}_l, \mathbf{C}_j)$.
- Since $(\mathbf{C}_i \perp \mathbf{C}_j|\mathbf{C}_l)$ we can compute

$$\hat{P}_\Phi(\mathbf{C}_i, \mathbf{C}_j) = \sum_{\mathbf{C}_l - \mathbf{C}_i} \hat{P}_\Phi(\mathbf{C}_i, \mathbf{C}_l)\hat{P}_\Phi(\mathbf{C}_j|\mathbf{C}_l)$$

Constructing the Clique Tree

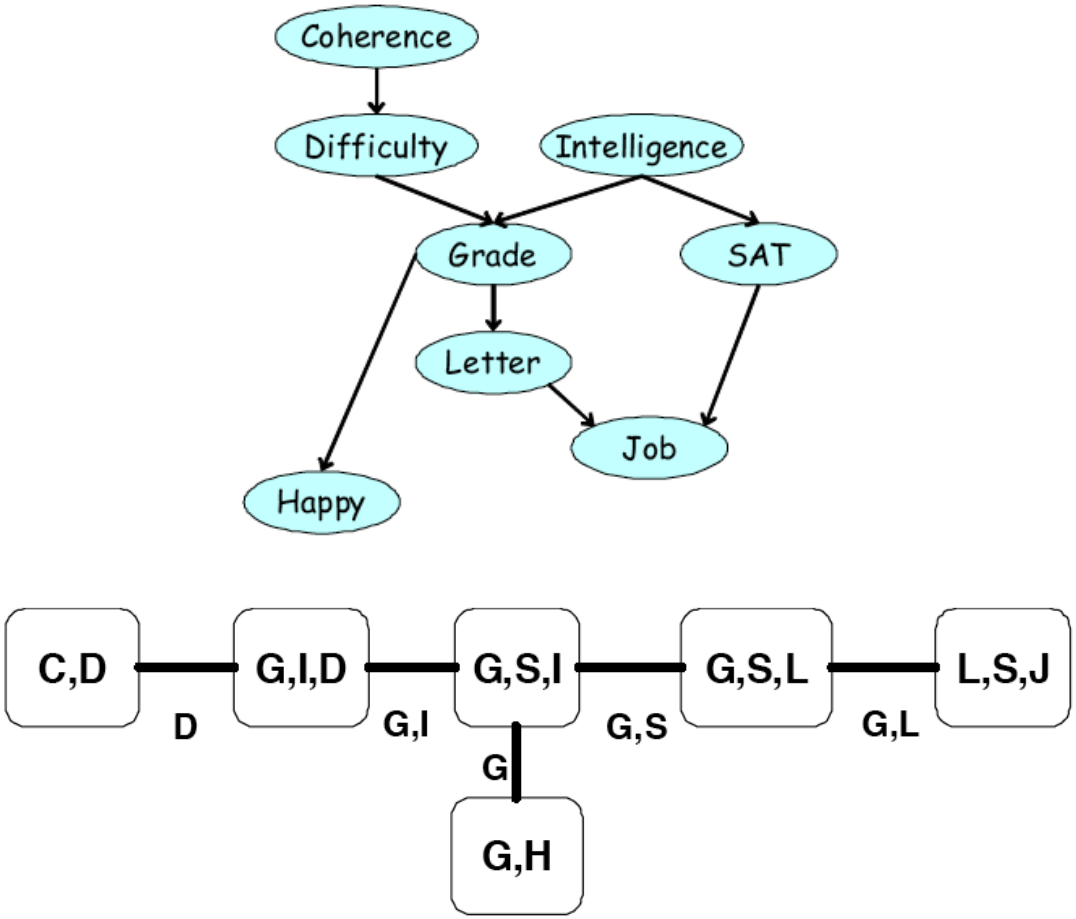
Two basic approaches

1. From Variable Elimination
2. Direct graph manipulation

Clique tree from VE

- An execution of VE can be associated with a cluster graph.
- A cluster $\mathbf{C}_i$ corresponds to the factor ψ_i generated during the execution of the algorithm.
- An undirected edge connects $\mathbf{C}_i$ and $\mathbf{C}_j$ when τ_i is used in the computation of ψ_j.
- This cluster graph is a tree and satisfies the running intersection property, hence it's a clique tree.
- Each factor in an execution of VE with an ordering $\prec$ is a subset of a clique in the induced graph $\mathcal{I}_{\Phi,\prec}$.
- Every maximal clique is a factor in the computation.
- It is standard to reduce the tree to only contain maximal cliques by removing it and connecting to the neighbors of the removed clique.

Example



(Elimination order J, L, S, H, C, D, I, G)

Clique tree from chordal graph

- We proved that the induced graph $\mathcal{I}_{\prec,\Phi}$ is a chordal graph.
- Any chordal graph can be used as the basis for inference.
- We construct a chordal graph $\mathcal{H}^*$ that contains the edges in $\mathcal{H}_\Phi$. Finding minimum triangulation is NP hard. Heuristics used.
- We find the maximal cliques and we connect them: finding the maximal cliques is NP hard.
  - For chordal graphs we can use the maximum cardinality and collect the maximum cliques generated this way.
  - Start with a family each of which is guaranteed to be a clique and do greedy search connecting nodes until it no longer induces a fully connected subgraph.
  - Connect the edges by maximum cardinality or maximum spanning tree.

Junction tree algorithm

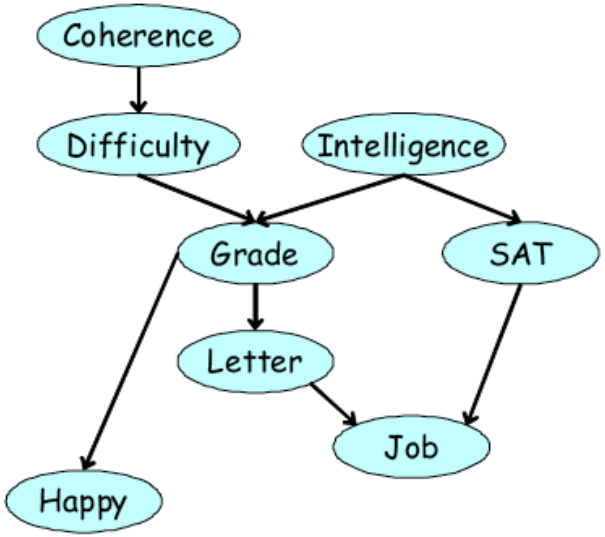
Clique trees are called **junction trees**

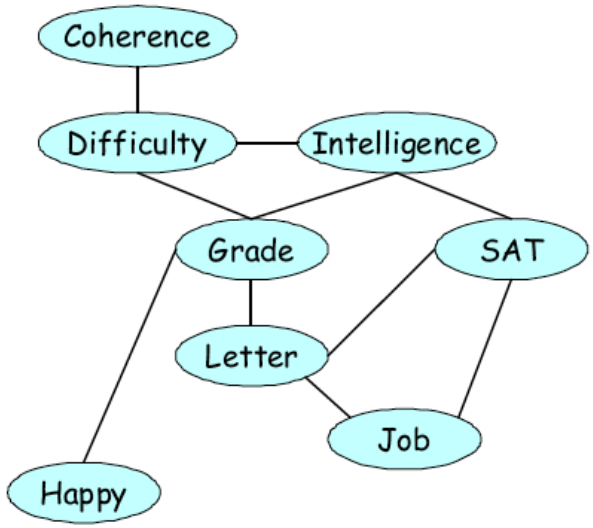
- If the graph is a BN then moralize it.
- Introduce the evidence.
- Triangulate the graph to make it chordal.
- Construct a junction tree (clique tree) from the triangulated graph.
- Propagate the probabilities using message passing, via sum-product BP or belief update.

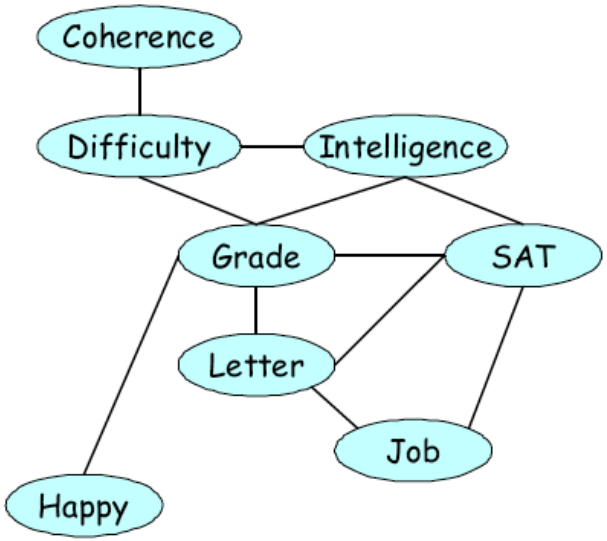
For graphs of large treewidth this is very inefficient.

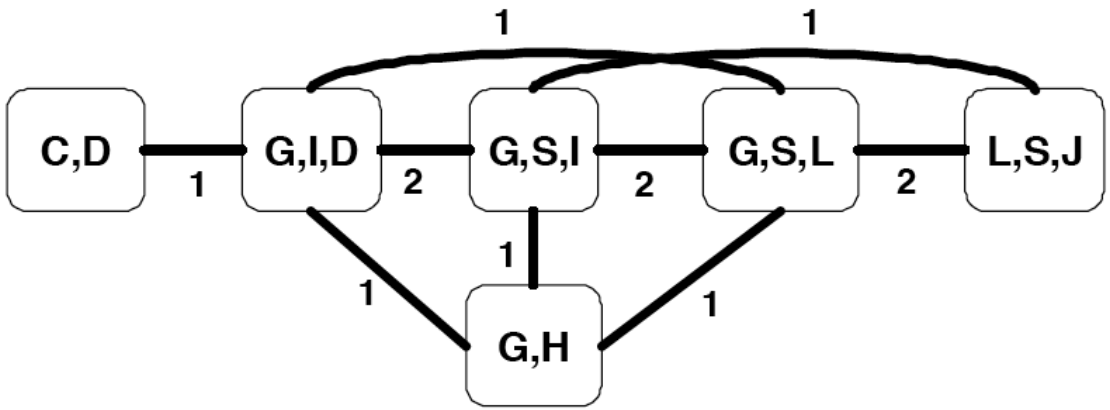
The **treewidth** measures the number of graph vertices mapped onto any tree node in an optimal tree decomposition.

Example


Coherence
Difficulty
Intelligence
Grade
SAT
Letter
Job
Happy


Coherence
Difficulty
Intelligence
Grade
SAT
Letter
Job
Happy


Coherence
Difficulty
Intelligence
Grade
SAT
Letter
Job
Happy


C,D
G,I,D
G,S,I
G,S,L
L,S,J
G,H
1
1
1
2
2
2
1
1
1