1 The Junction Tree Algorithm (Hugin algorithm) Chapter 16(17?), Jordan Notes by Broňa Brejová

1.1 Goals of the algorithm

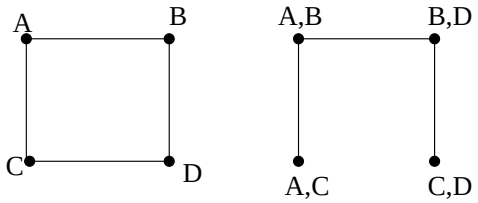
Given: directed or undirected model in G with set of variables V, set of evidence variables $E \subseteq V$ and their values $\overline{x}_E$.

Goal: For every clique C compute $p(x_{C \setminus E} | \overline{x}_E)$.

Clique tree: Tree in which nodes are all maximal cliques of G and they are somehow connected by edges to form a tree. Add a special node called separator to the middle of each edge (C_1, C_2) and label it with $C_1 \cap C_2$ (this is also a clique, but not maximal).

Junction tree: A clique tree is a junction tree if for each vertex v the set of cliques containing v is a connected subgraph of the tree.

Not every graph has a junction tree:



Alternative representation of the joint probability:
Originally:

$$p(x) = \frac{1}{Z} \prod_C \psi_C(x_C)$$

New form:

$$p(x) = \frac{1}{Z} \frac{\prod_C \psi^{**}_C(x_C)}{\prod_S \phi^{**}_S(x_S)}$$

Desired property (**local consistency**): for each two adjacent cliques U and W and their separator $S = U \cap W$:

$$\sum_{x_{U \setminus S}} \psi^{**}_U(x_S, x_{U \setminus S}) = \sum_{x_{W \setminus S}} \psi^{**}_W(x_S, x_{W \setminus S}) = \phi^{**}_S(x_S)$$

We will write such equations in a shortened form:

$$\sum_{U \setminus S} \psi^{**}_U = \sum_{W \setminus S} \psi^{**}_W = \phi^{**}_S$$

Later we will prove that if local consistency is true for every edge of a junction tree, then for every clique C and every separator S:

$$p(x_C) \propto \psi_C^{**}$$
$$p(x_S) \propto \phi_S^{**}$$

(i.e., we get our desired marginals).

1.2 The Hugin algorithm (s. 16.11)

- **Moralization:** directed graph $\rightarrow$ undirected graph
- **Introduction of evidence:** representation of $p(x_{V\setminus E}, x_E) \rightarrow$ representation of $p(x_{V\setminus E}|x_E)$.
- **Triangulation:** graph $\rightarrow$ graph that has a junction tree.
- **Construction of the junction tree:** find cliques, separators, initialize $\phi_S = 1$.
- **Propagation of probabilities:** achieve local consistency for every edge.

1.3 Propagation of probabilities (s. 16.5, 16.6)

Let us assume that we already have an undirected graph with a junction tree (and no observed variables). We want to manipulate local functions ψ and ϕ so that the joint probability stays the same

$$p(x) = \frac{1}{Z} \frac{\prod_C \psi_C(x_C)}{\prod_S \phi_S(x_S)}$$

and local consistency holds for each edge

$$\sum_{U\setminus S} \psi_U = \sum_{W\setminus S} \psi_W = \phi_S$$

Function UPDATE(U, W)**:**

$$\phi_S^* = \sum_{U\setminus S} \psi_U$$
$$\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W$$
$$\psi_U^* = \psi_U$$

Lemma 1. *Function UPDATE satisfies the following properties*

1. *The joint probability remains the same (i.e. $\psi_W \psi_U / \phi_S = \psi_W^* \psi_U^* / \phi_S^*$).*
2. *$\sum_{U\setminus S} \psi_U^* = \phi_S^*$.*
3. *If $\sum_{W\setminus S} \psi_W = \phi_S$, then also $\sum_{W\setminus S} \psi_W^* = \phi_S^*$*

Proof. The first property was proved in the last class. The second property clearly holds from definition of ϕ_S^* and ψ_U^*.

$$\begin{aligned}\sum_{W\setminus S} \psi_W^* &= \sum_{W\setminus S} \frac{\phi_S^*}{\phi_S}\psi_W \\ &= \frac{\phi_S^*}{\phi_S}\sum_{W\setminus S}\psi_W \\ &= \frac{\phi_S^*}{\phi_S}\phi_S \\ &= \phi_S^*\end{aligned}$$

□

Corollary 1. *After UPDATE(U, W) and UPDATE(W, U) the local consistency holds for U and W.*

Message-passing protocol. How to do UPDATE's in a junction tree with more than 2 vertices? Root the tree in an arbitrary node, then first send messages from the leaves towards the root and then from the root back towards the leaves.

```
function Main:
CollectEvidence(root)
DistributeEvidence(root)

function CollectEvidence(node W)
  for each child U of node
     CollectEvidence(U)
     Update(U, W)

function DistributeEvidence(node W)
  for each child U of node
     Update(W, U)
     CollectEvidence(U)
```

It is easy to see that if U is a child of W and S is their separator, then after CollectEvidence(W) finishes, $\sum_{U\setminus S}\psi_U^* = \phi_S^*$. Recursion then continues in other parts of the tree, but they do not change U nor S, therefore this holds until we execute DistributeEvidence(W). At this point part 3 of the lemma applies and therefore after UPDATE(W, U) both U and W will marginalize correctly. Since that point we will not change neither U nor W.

1.4 Final proof: If local consistence holds in a junction tree then local functions are marginal probabilities (s. 16.7)

Theorem 1. *Let probability $p(x)$ be represented by the clique potentials ψ_C and separator potential ϕ_S. Then if the local consistence holds for each edge in the juction tree, then clique and separator*

potentials are proportional to local marginal probabilities:

$$\psi_C \propto p(x_C)$$

$$\phi_S \propto p(x_S)$$

Proof. The separators are subsets of cliques. By local consistency, if clique potentials are proportional to marginals, then so are separators. Therefore we only need to prove the results for clique potentials.

The proof by induction in the number of cliques. For a single clique it holds by the definition of $p(x) = \phi_C(x)/Z$. Let us suppose that the results holds for junction trees with N cliques and consider a junction tree with $N+1$ cliques.

Let C be a leaf in the juction tree. Let S be the separator for this node, and let $R = C \setminus S$, $T = V \setminus C$. Sets R, S and T are disjoint and their union is V. Vertices in R do not occur in any other clique except C by junction tree property. Therefore S separates R from the rest of the tree and $R \perp T|S$.

$$\begin{aligned}
p(x_S, x_T) &= \sum_R p(x) \\
&= \frac{1}{Z} \sum_R \frac{\prod_{C'} \psi_C(x_C)}{\prod_{S'} \phi_S(x_S)} \\
&= \frac{1}{Z} \sum_R \frac{\psi_C(x_C) \prod_{C' \neq C} \psi_{C'}(x_{C'})}{\phi_S(x_S) \prod_{S' \neq S} \phi_{S'}(x_{S'})} \\
&= \frac{1}{Z} \cdot \frac{\sum_R \psi_C(x_C)}{\phi_S(x_S)} \cdot \frac{\prod_{C' \neq C} \psi_{C'}(x_{C'})}{\prod_{S' \neq S} \phi_{S'}(x_{S'})} \\
&= \frac{1}{Z} \cdot \frac{\prod_{C' \neq C} \psi_{C'}(x_{C'})}{\prod_{S' \neq S} \phi_{S'}(x_{S'})}
\end{aligned}$$

This shows that $p(x_S, x_T)$ is represented by clique potentials and separator potentials on the junction tree over $S \cup T$. By the induction hypothesis the clique potential on this tree are equal to marginals.

It remains to show that the clique potential on C is a marginal. The neighbour of C in the tree is a marginal and since S is locally consistent with this neighbour, $\phi_S(x_S) \propto p(x_S)$.

Since $R \perp T|S$, we have that $p(x_R|x_S, x_T) = p(x_R|x_S)$ and therefore

$$p(x) = p(x_R|x_S, x_T)p(x_S, x_T) = p(x_R|x_S)p(x_S, x_T)$$

Therefore

$$\begin{aligned}
p(x_R|x_S) &= \frac{p(x)}{p(x_S, x_T)} &(1)\\
&= \frac{\frac{1}{Z} \cdot \frac{\prod_C \psi_C(x_C)}{\prod_S \phi_S(x_S)}}{\frac{1}{Z} \cdot \frac{\prod_{C' \neq C} \psi_{C'}(x_{C'})}{\prod_{S' \neq S} \phi_{S'}(x_{S'})}} &(2)
\end{aligned}$$

$$= \frac{\psi_C(x_C)}{\phi_S(x_S)} \tag{3}$$

$$\propto \frac{\psi_C(x_C)}{p(x_S)} \tag{4}$$

$$\tag{5}$$

Therefore $\psi_C(x_C) \propto p(x_R|x_S)p(x_S) = p(x_C)$. □

1.5 Moralization (s. 16.2)

- For each node X_i connect all parents of X_i to a clique
- Drop orientation of edges
- For each node X_i multiply $p(x_i|x_{\pi_i})$ onto potential of a maximal clique containing $\pi_i \cup \{X_i\}$ (choose one of possibly several such cliques).

The new undirected model represents the same distribution.

1.6 Introduction of evidence (s. 16.3)

Let $H = V \setminus E$. Assume $\overline{x}_E$ is fixed (evidence).

$$\tilde{\psi}_{C\cap H}(x_{C\cap H}) \stackrel{def}{=} \psi_C(\underbrace{x_{C\cap H}, \overline{x}_{C\cap E}}_{x_C})$$

This corresponds to taking a slice of the local function.

Example:

$$\psi_{\{X,Y\}} = \begin{bmatrix} 0.12 & 0.08 \\ 0.24 & 0.56 \end{bmatrix}$$

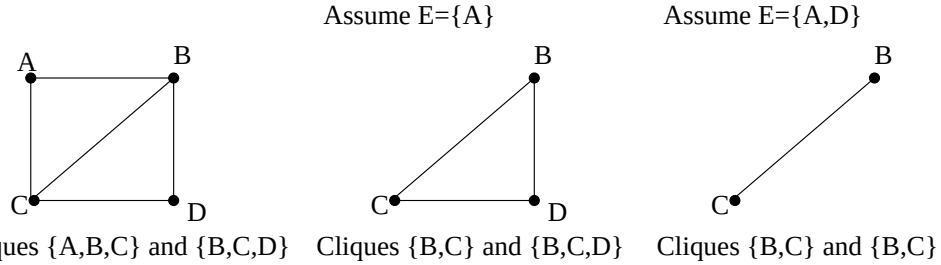
If $E = \{Y\}$ and $\overline{y} = 1$, we get

$$\tilde{\psi}_Y = \begin{bmatrix} 0.08 \\ 0.56 \end{bmatrix}$$

$$\begin{aligned} p(x_H|\overline{x}_E) &= \frac{p(x_H, \overline{x}_E)}{p(\overline{x}_E)} \\ &= \frac{\frac{1}{Z}\prod_C \psi_C(x_{C\cap H}, \overline{x}_{C\cap E})}{\sum_H \frac{1}{Z}\prod_C \psi_C(x_{C\cap H}, \overline{x}_{C\cap E})} \\ &= \frac{\prod_C \tilde{\psi}_{C\cap H}(x_{C\cap H})}{\underbrace{\sum_H \prod_C \tilde{\psi}_{C\cap H}(x_{C\cap H})}_{Z'}} \\ &= \frac{1}{Z'}\prod_C \tilde{\psi}_{C\cap H}(x_{C\cap H}) \end{aligned}$$

$\tilde{\psi}_{C\cap H}(x_{C\cap H})$ are local functions associated with cliques $C \cap H$ of induced subgraph $G[H]$. Each maximal clique in $G[H]$ is in fact in set of cliques $\{C \cap H | C \text{ is a clique in } G\}$, but we may get cliques that are not maximal or $C_1 \cap H = C_2 \cap H$ for some $C_1 \neq C_2$. We just multiply together all local function associated with one maximal clique and its subcliques.

Example:



So for given graph G, set of local functions ψ_C representing $p(x_H, x_E)$, and set of evidence variables E we have obtained a new graph G', new set of local functions $\tilde{\psi}'_C$ that represents $p(x_H|x_E)$.

So from now on we do not need to worry about evidence E – instead of computing $p(x_{C\cap H}|\overline{x}_E)$ in the old graph we compute $p(x_{C\cap H})$ in the new graph.

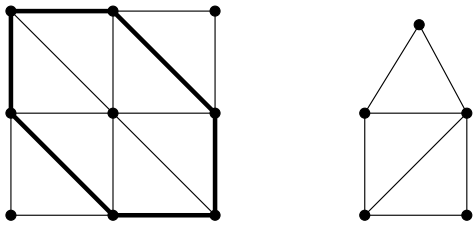
- In the book some places consider the evidence in the later parts of the algorithm and some do not.
- I am not sure whether it is good in practice to transform the graph in this way.
- Anyway, we will do all proofs without evidence and you can figure out yourselves how the evidence will change it.

1.7 Triangulation (s. 16.8, 16.14.2 (appendix))

- Not every graph has a junction tree
- The rest of the algorithm only works for the junction tree
- Given graph G we add some edges to get a new graph G' that has a junction tree.
- By adding edges we change the set of maximal cliques, but again each local function can be multiplied onto one of the cliques for which it is subset.

Definition 1. *A* chord *of a cycle in a graph is edge connecting two vertices not adjacent in the cycle. A cycle is* chordless *if it has no chords. A graph is* triangulated *(also called* chordal*), if no cycle of length at least 4 is chordless.*

Example:



Theorem 2. *A graph has a junction tree if and only if it is triangulated.*

We will prove $\Leftarrow$ direction only, needed to prove correctness of the algorithm.

Observation: Let $A \subseteq V$ and let $G[A]$ is subgraph induced by A, i.e. $G[A] = (A, E \cap A^2)$. If G is triangulated, then so is $G[A]$.

Lemma 2. *Let $G = (V, E)$ be a non-complete triangulated graph with at least three nodes. Then there exists a decomposition of V into non-empty disjoint sets A, B, and S such that S is complete and it separates A and B (i.e. every path from a vertex in A to a vertex in B goes through a vertex in S).*

Proof. If graph not connected, choose S as a single vertex and each of A and B contain some connected components.

Choose a pair of non-adjacent nodes α and β. Let S be the minimal set of nodes such that any path from α to β passes through S. Let A be the set of nodes reachable from α when S is removed, B be the set of nodes reachable from β when S is removed and let $X = V \setminus (A \cup B \cup S)$. Note that X may be empty. Clearly $A \cup X$ and B are separated by S. We only need to prove that S is complete.

Let C and D be non-adjacent nodes in S. Both C and D have neighbours in both A and B, because S is minimal. Neighbour of C in A and neighbour of D in A are connected in A (at least through α). Take the shortest path from C to D that uses only vertices in A as internal vertices. Similarly take the shortest path from C to D that uses only vertices in B as internal vertices. Joins these paths – you get cycle of length at least 4. It must have a chord. This chord cannot be edge (C, D), because they are non-adjacent. The cord cannot be entirely in A because the path was chosen as shortest. It cannot be from a vertex in A to C or D for the same reason. By symmetry the same holds for B. It cannot go from A to B because then S is not a separator. Contradiction. □

Definition 2. *A node is* simplicial *if all of its neighbours are connected.*

Lemma 3. *Every triangulated graph that contains at least two nodes has at least two simplicial nodes. If the graph is not complete, then these nodes can be chosen to be non-adjacent.*

Proof. We use induction on the number of vertices. As a base case the proof holds for the graph with two vertices.

Consider triangulated graph with $N+1$ vertices. If complete, every node is simplicial. Otherwise we decompose the graph into A, B and S as in the previous lemma. Subgraph $G[A \cup S]$ is triangulated and by induction hypothesis it has at least two simplicial nodes. If $G[A \cup S]$ is not complete, choose these two nodes non-adjacent. Then at least one of them is in A (S is complete).

This node is simplicial in $G[A \cup S]$ and it has no neighbours in B, therefore it is simplicial in G as well. If $G[A \cup S]$ is complete, choose any vertex in A – it is simplicial in G. By symmetry we can find simplicial node in B. These two nodes are non-adjacent. □

Lemma 4. *Every triangulated graph has a junction tree.*

Proof. Again use induction and base case $|V| = 1$ is trivial. Consider graph G with $N + 1$ nodes. It has a simplicial node α. We remove α, getting an induced subgraph $G' = G[V \setminus \{\alpha\}]$ which is triangulated. By induction hypothesis graph G' has a junction tree T.

Let C by the clique formed by α and its neighbours, and let $C' = C \setminus \{\alpha\}$. C is maximal clique in G because no other vertex is connected to α. It is also the only maximal clique in G containing α.

If clique C' is a node in T, we add α to this node to get a junction tree for G.

If C' is not a node in T, then it most be a subset of at least one maximal clique D. Add C as a new node to T, connected as a leaf to clique D. The resulting tree is a junction tree because (1) α is only in clique C and (2) all other vertices in C are in D as well. □

Note: The proof implies that the number of maximal cliques in a triangulated graph is at most $|V|$. In general graph it can be exponential (consider a graph that is a complement to a path of length n).

Algorithm for triangulation: If we triangulate the graph (add edges until it is triangulated), we get a graph with a junction tree. However how to triangulate the graph? One way is to use elimination algorithm from Chapter 3. Again we have to be careful about elimination orders.

```
Triangulate(G, elimination order of vertices O)
  if G has one vertex, return G
  let v be the first vertex in O
  form a clique from neighbours of v in G
  Triangulate(G-v,O-v)
```

Theorem 3. *The procedure described above yields a triangulated graph.*

Proof. By induction. Trivial for graph with 1 vertex. Consider graph with $N + 1$ nodes. We eliminate node and by induction the new graph with N nodes is triangulated. By adding the new node v we do not create any chordless cycles because all neighbours are connected. □

Jensen 1996: if a graph is triangulated there exists an ordering such that elimination algorithm does not introduce any new edges.

1.8 Construction of the junction tree (s. 16.9, 16.10)

We know that the triangulated graph has a junction tree and the proof of Lemma 4 has a recursive procedure for constructing it by eliminating a simplicial vertex, constructing the tree for the rest of the graph and then adding the vertex either to some existing node or creating a new leaf.

Different construction: Find all maximal cliques, create a graph with cliques as nodes and size of separator as a weight of an edge, find maximum weight spanning tree (Prim's or Kruskal's algorithm). This tree is a junction tree.

Why? Consider variable X_k. If it is included in j maximal cliques, it will be in at most $j-1$ separators in any clique tree. In addition, this number will be $j-1$ only if the cliques containing X_k form a connected subgraph of the tree. Therefore the junction tree has the largest possible sum of separator sizes over all spanning trees.

1.9 Computational complexity (s. 16.13)

- **Moralization:** $O(N^2)$ + manipulation of tables
- **Introduction of evidence:** $O(N+M)$ + manipulation of tables
- **Triangulation:** NP-hard to find optimal triangulation, but can be done heuristically in polytime + manipulation of tables
- **Construction of the junction tree:** $O(N^2)$ using maximum spanning tree, or $O(N^3)$ recursively.
- **Propagation of probabilities:** $O(N)$ messages passed, each needs table manipulation

Size of a table is r^k where r is the number of values per variable, k is the size of a clique. The main problem is to keep cliques small (in triangulation phase).