JUNCTION TREE ALGORITHM

Tuesday, September 9, 2008
Rice University
STAT 631 / ELEC 639: **Graphical Models**

Scribe:
David Kahle
Terrance Savitsky
Stephen Schnelle

Instructor:
Dr. Volkan Cevher

1. Introduction

In the previous scribes, we introduced general objects used in graphical models as well message passing schemes such as the sum-product algorithm and the max-sum algorithm for efficient computation of local marginals over subsets of the graph as well as the most probable state of the graph. For graphs which are trees, we noted that the algorithms are exact. In this article, we discuss a more flexible tool known as the junction tree algorithm which is used to compute local marginals of subsets of arbitrary graphs and yet still retains the property of being exact. The article draws largely from the exposition contained in Bishop[2], Lauritzen[4], Barber[1], and Wainwright and Jordan[5].

2. The Junction Tree Algorithm

The junction tree algorithm comprises 7 steps, listed below, which are expounded in the 7 subsections of this section.

(1) Moralize (directed graphs only)
(2) Triangulate
(3) Form the junction tree.
(4) Assign the potentials to the junction tree cliques and initialize the separator potentials to unity
(5) Select an (arbitrary) root clique
(6) Carry out message passing with absorption to and from the root clique until updates passed along both directions of every link on the junction tree.
(7) Read off the clique marginal potentials from the junction tree

2.1. Moralizing the graph (Directed graphs only).

For uniform applicability, directed graphs require the additional step, called "moralization", in order to be converted into an undirected graph. Thus, to perform the junction tree algorithm on an already undirected graph, one proceeds directly to step (2). The procedure described in this section is only necessary if we begin with a directed graph.

Moralization, the transformation of the directed graph $\vec{\mathcal{G}} = (\boldsymbol{N}_{\vec{\mathcal{G}}}, \mathcal{E}_{\vec{\mathcal{G}}})$ to an undirected graph $\mathcal{G} = (\boldsymbol{N}_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}})$ plays an important role in the junction tree algorithm. Informally, moralizing entails adding edges between parents of nodes and dropping the directions. More formally, the transformation from $\vec{\mathcal{G}}$ to $\mathcal{G}$ requires the addition (to $\mathcal{E}_{\vec{\mathcal{G}}}$) of two sets, $\mathcal{E}_{\vec{\mathcal{G}}\to\mathcal{G}}$ and $\mathcal{E}_{\vec{\mathcal{G}}\leftrightarrow\vec{\mathcal{G}}}$. The former joins the parents, and the latter "drops" the directions (by adding edges in the reverse direction). They are properly defined

$$\begin{aligned}
\mathcal{E}_{\vec{\mathcal{G}}\to\mathcal{G}} &:= \left\{(X_i, X_j) \in \boldsymbol{N}^2 : \exists X_k \in \boldsymbol{N} \ \ni: \ (X_i, X_k) \in \mathcal{E}_{\vec{\mathcal{G}}} \text{ and } (X_j, X_k) \in \mathcal{E}_{\vec{\mathcal{G}}}\right\} \\
\mathcal{E}_{\vec{\mathcal{G}}\leftrightarrow\vec{\mathcal{G}}} &:= \left\{(X_j, X_k) \in \boldsymbol{N}^2 : (X_j, X_k) \notin \mathcal{E}_{\vec{\mathcal{G}}} \text{ but } (X_k, X_j) \in \mathcal{E}_{\vec{\mathcal{G}}}.\right\}
\end{aligned}$$

Thus, the moralization step is simply $\mathcal{E}_{\vec{\mathcal{G}}} \cup \mathcal{E}_{\vec{\mathcal{G}}\to\mathcal{G}} \cup \mathcal{E}_{\vec{\mathcal{G}}\leftrightarrow\vec{\mathcal{G}}}$, and the moralized graph is

$$\mathcal{G} := (\boldsymbol{N}_{\vec{\mathcal{G}}}, \mathcal{E}_{\vec{\mathcal{G}}} \cup \mathcal{E}_{\vec{\mathcal{G}}\to\mathcal{G}} \cup \mathcal{E}_{\vec{\mathcal{G}}\leftrightarrow\vec{\mathcal{G}}}). \tag{1}$$

Example 1. Suppose we are presented with the directed graph $\vec{\mathcal{G}} = \left\{\boldsymbol{N}_{\vec{\mathcal{G}}}, \mathcal{E}_{\vec{\mathcal{G}}}\right\}$ where

$$\begin{aligned} \boldsymbol{N}_{\vec{\mathcal{G}}} &:= \{A, B, C, D, E, F, G\} \\ \mathcal{E}_{\vec{\mathcal{G}}} &:= \{(A, C), (A, D), (B, D), (C, F), (D, F), (D, G), (E, G)\}. \end{aligned}$$

A pictorial representation of $\vec{\mathcal{G}}$ is contained in Figure 1. To moralize the graph, we join the parents and remove all directions. Here, A and B are both parents of D, so the edges (A, B) and (B, A) must be added to $\mathcal{E}_{\vec{\mathcal{G}}}$ in the moralization step. Similarly, (C, D), (D, C), (D, E), (E, D) all need to be added. All of these are part of the joining step, that is, in $\mathcal{E}_{\vec{\mathcal{G}}\to\mathcal{G}}$. Thus,

$$\mathcal{E}_{\vec{\mathcal{G}}\to\mathcal{G}} = \{(A, B), (B, A), (C, D), (D, C), (D, E), (E, D)\}.$$

All that is left to do is remove the directions. This is done by adding all the reverses of the edges which aren't in the graph. For example, $(A, C) \in \mathcal{E}_{\vec{\mathcal{G}}}$ but $(C, A) \notin \mathcal{E}_{\vec{\mathcal{G}}}$, so (C, A) needs to be added to $\mathcal{E}_{\vec{\mathcal{G}}}$. Similarly, $(D, A), (D, B), (F, C), (F, D)$, (G, D), and (G, E) should be added. The entire set is therefore simply the mirror image of $\mathcal{E}_{\vec{\mathcal{G}}}$,

$$\mathcal{E}_{\vec{\mathcal{G}}\leftrightarrow\vec{\mathcal{G}}} = \{(C, A), (D, A), (D, B), (F, C), (F, D), (G, D), (G, E)\}.$$

The undirected graph resulting from the moralization we simply call $\mathcal{G}$,[1]

$$\mathcal{G} := (\boldsymbol{N}_{\vec{\mathcal{G}}}, \mathcal{E}_{\vec{\mathcal{G}}} \cup \mathcal{E}_{\vec{\mathcal{G}}\to\mathcal{G}} \cup \mathcal{E}_{\vec{\mathcal{G}}\leftrightarrow\vec{\mathcal{G}}}). \tag{2}$$

where

$$\begin{aligned} \boldsymbol{N}_{\mathcal{G}} &:= \{A, B, C, D, E, F, G\} \\ \mathcal{E}_{\mathcal{G}} &:= \{(A, C), (A, D), (B, D), (C, F), (D, F), (D, G), (E, G), \\ &\quad (A, B), (B, A), (C, D), (D, C), (D, E), (E, D), \\ &\quad (C, A), (D, A), (D, B), (F, C), (F, D), (G, D), (G, E)\}. \end{aligned} \tag{3}$$

The undirected graph $\mathcal{G}$ made from the moralization of $\vec{\mathcal{G}}$ is displayed in Figure 2 with the edges added by moralization in dashed blue.

||

2.2. **Triangulating the graph.** The second step (first for already undirected graphs) is triangulation. To begin, we need to have a notion of a triangulated graph. An undirected graph $\mathcal{G}$ is said to be *triangulated* if and only if for every cycle of length $n \geq 4$ possesses a chord.[2] Thus, in triangulating the undirected graph $\mathcal{G}$ we are adding edges so that the new graph $\mathcal{G}^{\triangle}$ is triangulated.[3] The

[1] This is instead of introducing new notation which makes explicit that it came from moralizing $\vec{\mathcal{G}}$. Wherever the distinction is ambiguous, we will use the notation $\mathcal{G}^m$.

[2] Recall that a n-cycle is a path with the same beginning and end; a chord is an edge joining a pair of nonconsecutive vertices in the cycle. Also recall that, by convention, the counting of cycle length begins with 0 as opposed to 1.

[3] In general, we will reserve the superscript $^{\triangle}$ on an undirected graph $\mathcal{G}$ to emphasize that $\mathcal{G}$ is triangulated.

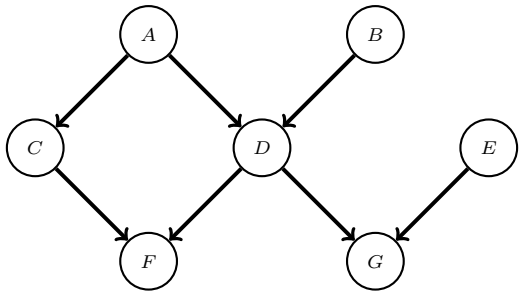

FIGURE 1. Original directed graph $\vec{\mathcal{G}}$ in Example 1

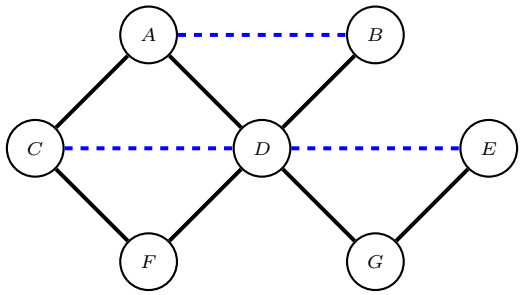

FIGURE 2. Undirected graph $\mathcal{G}$ in Example 1 resulting from the moralization of $\vec{\mathcal{G}}$ with moralization edges in dashed blue

process is not unique, as is demonstrated in Example 2; but the general idea is to continue the triangulation procedure (adding edges) until we obtain a triangulated graph and stop at that step.

Example 2. Consider the undirected graph $\mathcal{G} = (\boldsymbol{N}_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}})$ defined by

$$\begin{aligned} \boldsymbol{N}_{\mathcal{G}} &= (A, B, C, D, E) \\ \mathcal{E}_{\mathcal{G}} &= \{(A,B),(B,C),(C,D),(D,E),(E,A) \\ &= \ (B,A),(C,B),(D,C),(E,D),(A,E)\}, \end{aligned}$$

pictured in Figure 3. Any cycle, for example $A-B-C-D-E-A$, is of length 5 and possesses no chords. Thus, to begin we will add the undirected edge (edges, to speak precisely) (A,C) and (C,A) to $\mathcal{E}_{\mathcal{G}}$. The result is the graph $\mathcal{G}^{(1)} = (\boldsymbol{N}_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}^{(1)}})$ where

$$\begin{aligned} \mathcal{E}_{\mathcal{G}^{(1)}} &= \{(A,B),(B,C),(C,D),(D,E),(E,A),(A,C) \\ &= \ (B,A),(C,B),(D,C),(E,D),(A,E),(C,A)\}. \end{aligned}$$

The superscript $^{(1)}$ is used to denote that this graph is the result of the addition of one undirected edge; the point is that we no longer have $\mathcal{G}$, but a manipulated version of it. For our discussion, the number itself is not of any interest. Notice that only the edge set $\mathcal{E}_{\mathcal{G}}$ changes in each step of the triangulation process. The picture of $\mathcal{G}^{(1)}$ is contained in Figure 4 with the triangulation undirected edge in dashed red.

Upon inspection, the new graph, $\mathcal{G}^{(1)}$, is still not a triangulated graph since it exhibits, for example, the 4-cycle $A - C - D - E - A$; so we must continue adding edges. Thus, we will add the undirected edge (A, D) and (D, A). As before the result is $\mathcal{G}^{(2)} = (\boldsymbol{N}_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}^{(2)}})$ where

$$\begin{aligned} \mathcal{E}_{\mathcal{G}^{(2)}} &= \{(A,B),(B,C),(C,D),(D,E),(E,A),(A,C),(A,D) \\ &= \quad (B,A),(C,B),(D,C),(E,D),(A,E),(C,A),(D,A)\}. \end{aligned}$$

However, unlike its predecessors, $\mathcal{G}^{(2)}$ is triangulated. Thus, we set $\mathcal{G}^{\triangle} = \mathcal{G}^{(2)}$, completing the triangulation process.

||

Note that the triangulation procedure is not unique. The selection of which edges to add in Example 2 was completely arbitrary; we could have selected many different edges to add to triangulate $\mathcal{G}$. Which edges are of greatest interest to add is a question which will not be examined in this article.

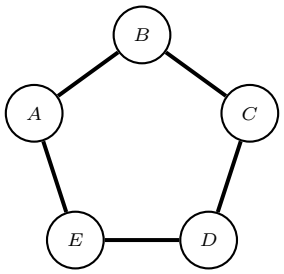

FIGURE 3. The untriangulated graph $\mathcal{G}$ in Example 2

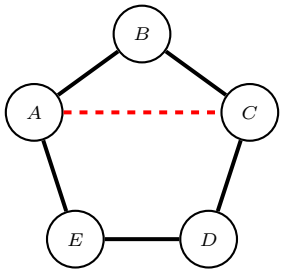

FIGURE 4. The graph $\mathcal{G}^{(1)}$ from Example 2 with the triangulation undirected edge in dashed red

2.3. **Forming the junction tree.** As in step (2), we begin with a few definitions. A *hypergraph* $\mathcal{H}$ is a collection of nonempty subsets of a finite set $\boldsymbol{H}$ (known as the *base set*). The elements of $\mathcal{H}$ (subsets of $\boldsymbol{H}$) are referred to as

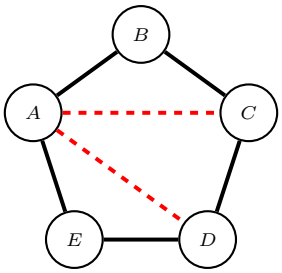

FIGURE 5. The graph $\mathcal{G}^{(2)} = \mathcal{G}^{\triangle}$ from Example 2 with the triangulation undirected edges in dashed red

hyperedges.[4] For example, if $\mathcal{G}$ is a finite undirected graph, the set of cliques of $\mathcal{G}$, denoted $\mathcal{C}(\mathcal{G})$, forms a hypergraph known as the *clique hypergraph*.

Recall that a tree $\mathcal{T} = (\boldsymbol{N}_{\mathcal{T}}, \mathcal{E}_{\mathcal{T}})$ is any connected, undirected graph without cycles and that the key property of such graphs is uniqueness of path between any two vertices. A junction tree is, not surprisingly, a particular kind of tree. Specifically, a *junction tree* $\mathcal{T}^{\mathfrak{J}} = (\mathcal{H}_{\mathcal{T}^{\mathfrak{J}}}, \mathcal{E}_{\mathcal{T}^{\mathfrak{J}}})$ is a tree whose nodes $\mathcal{H}_{\mathcal{T}^{\mathfrak{J}}}$ are a hypergraph (with respect to some base set) with the additional property that the intersection $\boldsymbol{U} \cap \boldsymbol{V}$ of any two nodes $\boldsymbol{U}, \boldsymbol{V} \in \mathcal{H}_{\mathcal{T}^{\mathfrak{J}}}$ is contained in every node $\boldsymbol{W}$ in the unique path joining $\boldsymbol{U}$ and $\boldsymbol{V}$. The property is referred to as the *running intersection property* or the *junction property*.

The definition of a junction tree is daunting at first because of the tiers of new definitions. Fortunately, a familiar example can help make the definition more tangible. It will also indicate why the formation of a junction tree is the third step in the algorithm.

Example 1 - Continued. Recall our graph achieved via moralization in Example 1 defined in (2) and (3) and pictured in Figure 2. Consider the clique hypergraph $\mathcal{C}(\mathcal{G})$. To be precise, to describe it we need to determine $\mathcal{H}$. The base set is of course $\boldsymbol{N}_{\mathcal{G}}$, and from the edge set $\mathcal{E}_{\mathcal{G}}$ we determine that

$$\mathcal{C}(\mathcal{G}) = \{\{A, C, D\}, \{F, C, D\}, \{D, A, B\}, \{G, D, E\}\}, \tag{4}$$

each set being a maximal clique of $\mathcal{G}$. Now, the definition of the junction tree specifies that such a hypergraph is thought of as the nodes of a tree with a special property. So, if we set $\mathcal{H}_{\mathcal{T}^{\mathfrak{J}}} = \mathcal{C}(\mathcal{G})$, the only thing that stands in the way of us and having a junction tree is the edge set of the tree, $\mathcal{E}_{\mathcal{T}^{\mathfrak{J}}}$, which can be anything as long as the graph which it generates is undirected, connected, contains no cycles, and satisfies the running intersection property (meaning that it has to be both a tree and satisfy the running intersection property).

[4]The definition of hypergraph is somewhat unfortunate. Intuitively, we would like a hypergraph to be a generalization of an undirected graph; however, precisely speaking, this is in fact not the case. The definition provided is closer to a generalization of what we have been referring to as the edge set of an undirected graph, $\mathcal{E}_{\mathcal{G}}$. A more appropriate definition would be an ordered pair $\mathcal{H} = (\boldsymbol{N}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}})$, where $\mathcal{E}_{\mathcal{H}}$ is a set of subsets (no longer restricted to pairs) of $\boldsymbol{N}_{\mathcal{H}}$. In this definition, a hypergraph $\mathcal{H}$ would be akin to the familiar notions of topological space and measurable space, the difference being that the set $\mathcal{E}_{\mathcal{H}}$ is not defined through a set of axioms. For the purposes of this article, however, the definition will be the one provided in the main body.

For ease of notation, let's write the maximal cliques as strings of letters instead of sets - this will be particularly helpful later. For example, we will call $ACD := \{A, C, D\}$, and similarly with the other cliques. An edge of a graph is simply an ordered pair, so an example of an element of $\mathcal{E}_{\mathcal{T}^\mathfrak{J}}$ would be (ACD, GDE). Again, since $\mathcal{H}_{\mathcal{T}^\mathfrak{J}}$ is fixed, we need only consider different possibilities for $\mathcal{E}_{\mathcal{T}^\mathfrak{J}}$. To ensure that $\mathcal{E}_{\mathcal{T}^\mathfrak{J}}$ defines an undirected graph, we just require that for each ordered pair in $\mathcal{E}_{\mathcal{T}^\mathfrak{J}}$, the ordered pair with the reverse ordering is also in $\mathcal{E}_{\mathcal{T}^\mathfrak{J}}$, so that takes care of one of the conditions and reduces the number of different possibilities. From here, we simply posit a few different possibilities.

$$\begin{aligned}
\mathcal{E}^{(1)}_{\mathcal{T}^\mathfrak{J}} &= \begin{Bmatrix}(FCD, DAB), (DAB, ACD), (DAB, GCD)\\ (DAB, FCD), (ACD, DAB), (GCD, DAB)\end{Bmatrix}\\
\mathcal{E}^{(2)}_{\mathcal{T}^\mathfrak{J}} &= \begin{Bmatrix}(FCD, DAB), (DAB, ACD), (ACD, GCD)\\ (DAB, FCD), (ACD, DAB), (GCD, ACD)\end{Bmatrix}\\
\mathcal{E}^{(3)}_{\mathcal{T}^\mathfrak{J}} &= \begin{Bmatrix}(GCD, DAB), (DAB, ACD), (ACD, FCD)\\ (DAB, GCD), (ACD, DAB), (FCD, ACD)\end{Bmatrix}
\end{aligned}$$

The pictorial representations of $\mathcal{E}^{(1)}_{\mathcal{T}^\mathfrak{J}}$, $\mathcal{E}^{(2)}_{\mathcal{T}^\mathfrak{J}}$, and $\mathcal{E}^{(3)}_{\mathcal{T}^\mathfrak{J}}$ are contained in Figures 6, 7, and 8. We now consider each individually.

Regarding $\mathcal{E}^{(1)}_{\mathcal{T}^\mathfrak{J}}$, we note that it is clearly fully connected with no cycles (and undirected), so all that needs to be checked is whether or not it satisfies the running intersection property. But, $FCD \cap GCD = \{C, D\} \nsubseteq DAB$ since C is not in the clique DAB. Thus, $\mathcal{E}^{(1)}_{\mathcal{T}^\mathfrak{J}}$ does not create a junction tree.

The second edge set, $\mathcal{E}^{(2)}_{\mathcal{T}^\mathfrak{J}}$, also creates a fully connected graph with no cycles, so we check the running intersection property. $FCD \cap ACD = \{C, D\} \nsubseteq DAB$, so it is also not a junction tree.

Again for $\mathcal{E}^{(3)}_{\mathcal{T}^\mathfrak{J}}$ all we need to check is the junction property.
$GDE \cap ACD = \{D\} \subseteq DAB$, so no problem there. $GDE \cap FCD = \{D\}$ which is contained in both DAB and ACD, so again we are safe. All that remains is to check DAB and FCD; so $DAB \cap FCD = \{D\} \subseteq ACD$. Thus, the edge set in fact generates a junction tree, which we will call $\mathcal{T}^\mathfrak{J} = (\mathcal{C}(\mathcal{G}), \mathcal{E}^{(3)}_{\mathcal{T}^\mathfrak{J}})$.

||

It turns out that to every triangulated graph corresponds at least one junction tree. Although we didn't have the ability to realize it before, the moralized graph $\mathcal{G}$ in Example 1 is in fact a triangulated graph, so step (2) required no addition of edges. The subsequent junction tree $\mathcal{T}^\mathfrak{J}$ is therefore just one example demonstrating a more general relationship between triangulated graphs and junction trees. The general strategy is to begin with the hypergraph $\mathcal{C}(\mathcal{G}^\triangle)$ defined by the maximal cliques of the moralized and triangulated graph $\mathcal{G}^\triangle$ and then search for an appropriate edge set which satisfies the properties of a junction tree. While the efficient determination of a such an edge set for an arbitrary triangulated graph is an important topic in its own right, for now we will be satisfied with the fact that we have found a junction tree and proceed from there.

There is one additional step which is taken when displaying the junction tree which makes explicit the running intersection property which will be so important in enforcing certain consistency properties later on in our discussion - that of introducing separator nodes. Displayed in the middle of each edge is placed a separator node which is conventionally boxed instead of circled, it contains the variables which are common to both of the nodes (the intersection). Thus, the junction tree we found in Example 2 is more commonly displayed as Figure 9; we will use this representation for the duration of the article.

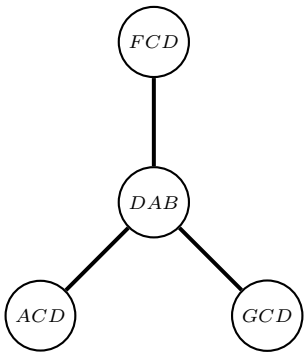

FIGURE 6. Graph on $\mathcal{C}(\mathcal{G})$ with edge set $\mathcal{E}^{(1)}_{\mathcal{T}^{\mathfrak{J}}}$

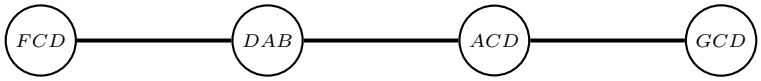

FIGURE 7. Graph on $\mathcal{C}(\mathcal{G})$ with edge set $\mathcal{E}^{(2)}_{\mathcal{T}^{\mathfrak{J}}}$

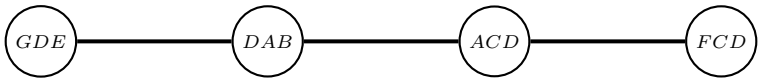

FIGURE 8. Graph on $\mathcal{C}(\mathcal{G})$ with edge set $\mathcal{E}^{(3)}_{\mathcal{T}^{\mathfrak{J}}}$

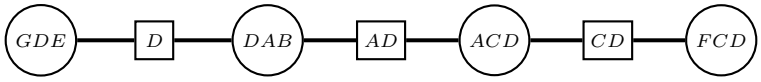

FIGURE 9. Junction tree $\mathcal{T}^{\mathfrak{J}}$ with separator nodes

2.4. **Assigning potentials and initializing.** From the Hammersley-Clifford theorem, we know that the joint density of all the variables in the original graph factors into an appropriately scaled product of potential functions over the maximal cliques. For the junction tree algorithm, the clique potentials are set to the original potentials over the undirected graph (or even more explicitly if they are known from a known directed graph structure, the conditional distributions themselves), just as they would be in the sum-product algorithm. The potentials for the separator nodes are set to unity.

2.5. **Selecting an arbitrary root node.** The previous steps of the algorithm have been used to set up the nodes/cliques in a way that is suitable to apply a message passing algorithm. Now we must select a root node to begin. Each link between nodes and separators will be used twice during message passing, once in each direction. This is done by propagating messages "up" from each leaf to a root and then in reverse from the leaves to the root. Although trees are usually represented with a vertical orientation, we will represent them sideways as in 9 to preserve space.

Example 1 - Continued. In Figure 10, we select node GDE as root.

||

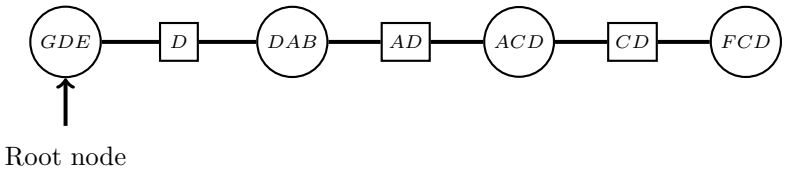

FIGURE 10. Junction tree $\mathcal{T}^{\mathfrak{J}}$ with node GDE emphasized as root

2.6. **Carrying out message passing.** Potentials were assigned in step (4). Because we have created a junction tree, there will be at least one node of the junction tree that is connected to only one neighbor and is not our arbitrarily chosen root of the tree. Do not choose the root as the idea is for our first pass through the tree, we pass towards the root. Pass the messages using the standard message passing algorithms for graphical modes. Select other nodes that are connected to only one neighbor, if present, and pass towards the root. Once an internal node of the tree (more than one neighbor) has received messages from all those nodes which it separates from the root, pass its updated message along towards the root. Finally reverse this process.

We can think of this in the sense of a traditional tree with a root node at the top; messages are passed up the tree at each stage, updating nodes along the path to the root with information from all of its children before moving up to the next level. Then with the updated root, we pass revised messages down the chain.

Fortunately, messages must be passed along the links in each direction only once. Forming the junction tree ensures that this algorithm will converge.

Example 1 - Continued. In Figure 11, passing messages are quite straightforward, up the tree and then back down.

||

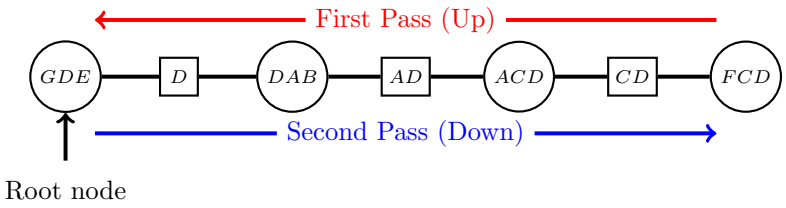

FIGURE 11. Message passing on the junction tree $\mathcal{T}^{\mathfrak{J}}$ with root GDE

2.7. **Evaluating desired marginal potentials.** After completing the bidirectional message passing scheme in step (6), the messages at each node correspond to the potential functions of each clique, which correspond to the joint probability distribution of the nodes in the clique. Because the nodes form a clique and thus are all connected, to find the probablity densities of a subset of these nodes (those the clique) we need only to marginalize the resulting potential function over the remaining variables.

Ideally, we would like to find a triangulated graph with minimal maximal clique size to make the computations of marginals inside each node nearly trivial. This provides us with at least some guidelines by which we can discriminate against different triangulations.

Example 1 - Continued. Suppose in our example our variable of interest is in fact just A. Then from Figure 12, we can marginalize over either nodes DAB or ACD to find the density of A. A consistency property exhibited by junction trees (to be discussed below) ensures that these two methods of obtaining the marginal of A are the same.

||

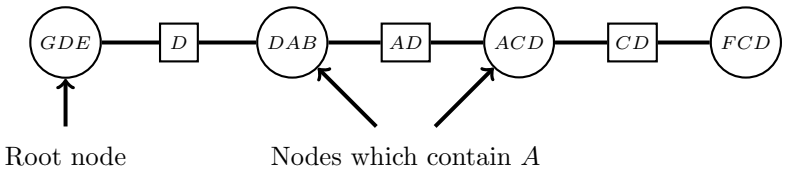

FIGURE 12. Junction tree $\mathcal{T}^{\mathfrak{J}}$ and nodes via which the marginal distribution of A can be acquired.

3. Consistency

The concept of consistency came into the discussion at the end of the junction tree algorithm. This concept is explored in more detain in this section in three parts - the idea, the importance, and the algorithm.

3.1. **Idea.** Given two adjacent nodes of the junction tree $\boldsymbol{V}$ and $\boldsymbol{W}$, we wish the potentials resulting from marginalizing over $\boldsymbol{V}$ or $\boldsymbol{W}$ to their separator, $\boldsymbol{S}$, to give the same potential for $\boldsymbol{S}$.

$$\sum_{\boldsymbol{w}\setminus\boldsymbol{s}} \Psi(\boldsymbol{w}) = p_{\boldsymbol{S}}(\boldsymbol{s}) = \sum_{\boldsymbol{v}\setminus\boldsymbol{s}} \Psi(\boldsymbol{v}); \quad p_{\boldsymbol{S}}(\boldsymbol{s}) = \Psi(\boldsymbol{s})$$

We call this condition of equality *consistency*.

Global consistency would imply that for any two nodes $\boldsymbol{V}$ and $\boldsymbol{W}$ with intersection $\boldsymbol{I}$, we have

$$\sum_{\boldsymbol{w}\setminus\boldsymbol{i}} \Psi(\boldsymbol{w}) = \sum_{\boldsymbol{v}\setminus\boldsymbol{i}} \Psi(\boldsymbol{v})$$

Hence the joint probability distribution of the intersection of any two nodes is the same whether marginalizing within either node, even if the two nodes are not neighbors.

3.2. **Importance.** Ensuring consistency is important. The idea of the junction tree algorithm and many other message passing schemes is that marginalization to find variables in a cluster need only be done over that cluster. Essentially each cluster is localized, and effects of other clusters are factored in during the message passing algorithm, but need not be considered later. However localization should require that if we look at a variable in the intersection of two (or more) cliques, we get the same results for its density whether we marginalize over one clique or another.

Fortunately consistency holds for a junction tree, under a variety of circumstances, including as data is observed and used to obtain better estimates of other parameter. This fact is outline in further detail below.

3.3. **Algorithm.** Suppose one or more variables in $\boldsymbol{V} = \boldsymbol{v}$ is observed (and clamped to a particular state). Ψ^* represents the updated potential function due to the observed data Our task is to modify $\Psi(\boldsymbol{w})$ and $\Psi(\boldsymbol{s})$ to satisfy:

$$\sum_{\boldsymbol{w}\setminus\boldsymbol{s}} \Psi^*(\boldsymbol{w}) = \Psi^*(\boldsymbol{s}) = \sum_{\boldsymbol{v}\setminus\boldsymbol{s}} \Psi^*(\boldsymbol{v})$$

Absorption replaces $\Psi^*(\boldsymbol{s})$ and $\Psi^*(\boldsymbol{w})$ with

$$\Psi^*(\boldsymbol{s}) = \sum_{\boldsymbol{v}\setminus\boldsymbol{s}} \Psi(\boldsymbol{v})$$

$$\Psi^*(\boldsymbol{w}) = \Psi(\boldsymbol{w}) \frac{\Psi^*(\boldsymbol{s})}{\Psi(\boldsymbol{s})}$$

Then,

$$\sum_{\boldsymbol{w}\setminus\boldsymbol{s}} \Psi^*(\boldsymbol{w}) = \sum_{\boldsymbol{w}\setminus\boldsymbol{s}} \Psi(\boldsymbol{w}) \frac{\Psi^*(\boldsymbol{s})}{\Psi(\boldsymbol{s})} = \frac{\Psi^*(\boldsymbol{s})}{\Psi(\boldsymbol{s})} \sum_{\boldsymbol{w}\setminus\boldsymbol{s}} \Psi(\boldsymbol{w})$$
$$= \frac{\Psi^*(\boldsymbol{s})}{\Psi(\boldsymbol{s})} \Psi(\boldsymbol{s}) = \Psi^*(\boldsymbol{s}) = \sum_{\boldsymbol{v}\setminus\boldsymbol{s}} \Psi^*(\boldsymbol{v})$$

and consistency is re-established. We say we have *absorbed* $\boldsymbol{v}$ into $\boldsymbol{W}$ through $\boldsymbol{S}$.

4. Example 2

A nice example is carried throughout [3] attributed to Lauritzen and Spiegelhalter. The following problem is set up, "Shortness-of breath (Dyspnoea) my be due to Tuberculosis, Lung cancer or Bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of Tuberculosis, while Smoking is known to be a risk factor for both Lung Cancer and Bronchitis. The results of a single X-ray do not discriminate between Lung Cancer and Tuberculosis, as neither does the presence or absence of Dyspnoea." They set up the graphical model as shown in Figure 13.

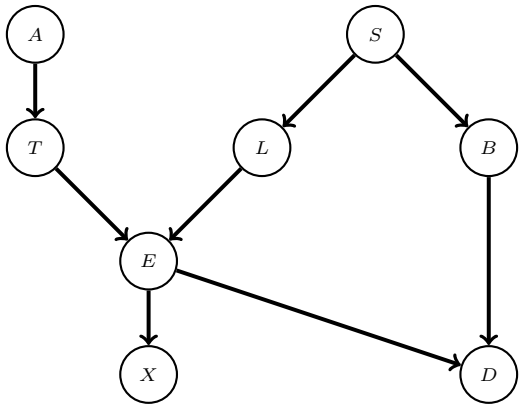

Figure 13. Original graphical model in Example 2

For the first step in the algorithm, we moralize the graph as shown in Figure 14. Next we triangulate the graph, as demonstrated in Figure 15. Then we form a corresponding junction tree, as in Figure 16. Separator nodes are added in Figure 17. Now, we need to select a root node, so we select SBL and start passing messages up to and down from SBL as in Figure 18 and Figure 19. From here, we could obtain any marginal we desire.

||

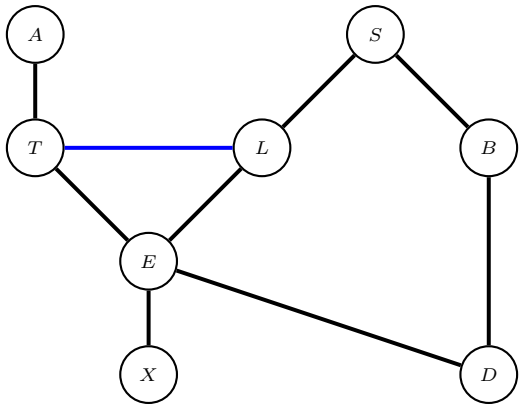

FIGURE 14. Moralized (in blue) graph in Example 2

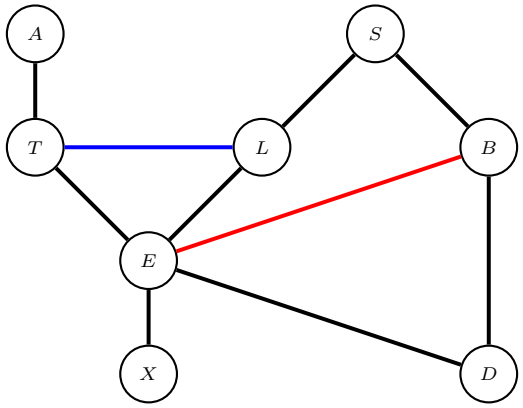

FIGURE 15. Triangulated (in red) moralized graph in Example 2

REFERENCES

1. D. Barber, *Probabilistic modelling and reasoning: The junction tree algorithm*, Working Paper (2004).
2. C. M. Bishop, *Pattern recognition and machine learning*, Springer, Oxford, 2007.
3. Robert Cowell, *Introduction in inference in bayesian networks*, (1999), 9–26.
4. SL Lauritzen, *Graphical Models. 1996*, New York.: Oxford University Press.
5. M.J. Wainwright and M.I. Jordan, *Graphical models, exponential families, and variational inference*, UC Berkeley, Dept. of Statistics, Technical Report **649** (2003).

RICE UNIVERSITY, DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, 6100 MAIN ST., HOUSTON, TEXAS, 77005.
E-mail address: dkahle@rice.edu, Terrance.D.Savitsky@rice.edu, srs2@rice.edu, volkan@rice.edu

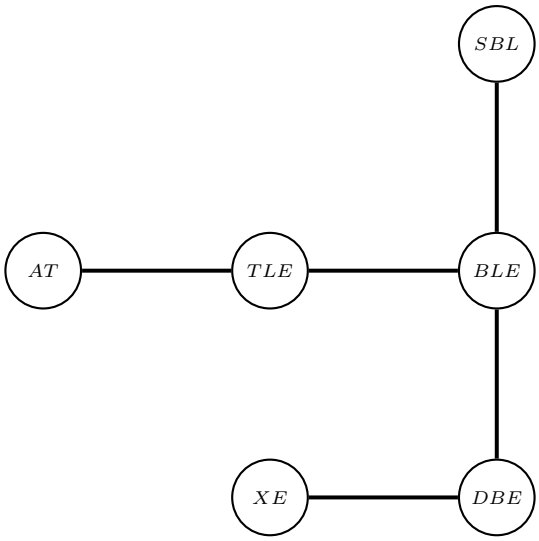

FIGURE 16. Junction tree of graph in Example 2

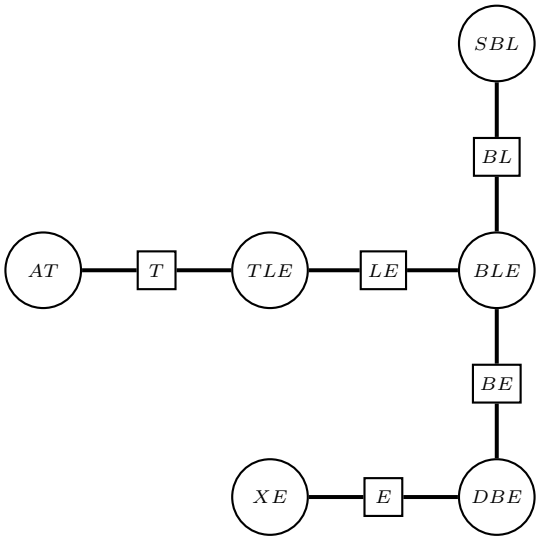

FIGURE 17. Junction tree of graph in Example 2 with separators (initialized to 1)

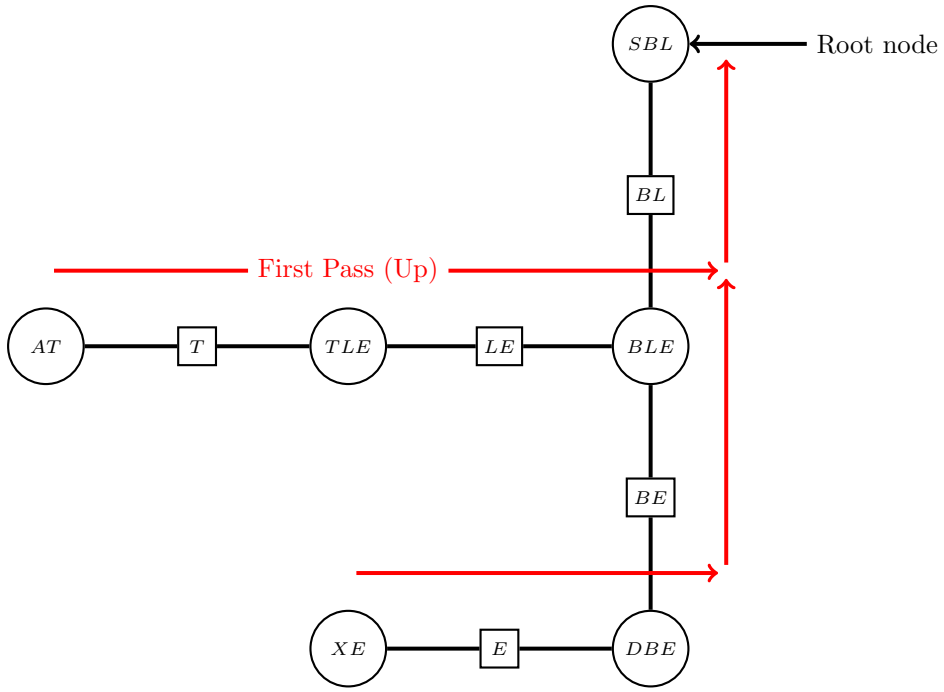

FIGURE 18. Passing messages up the junction tree

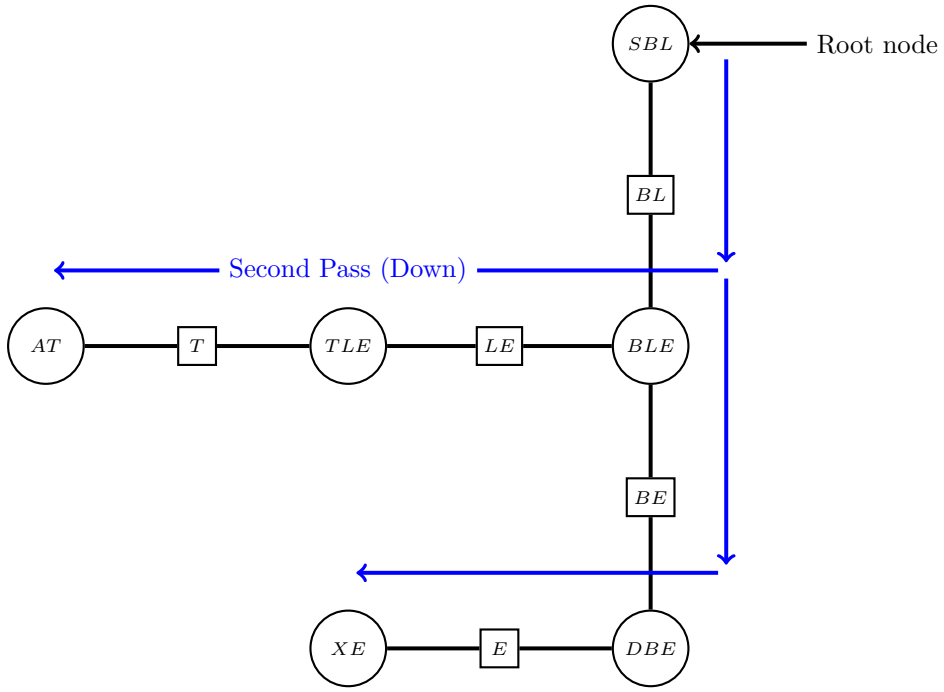

FIGURE 19. Passing messages down the junction tree