Exact Inference: Junction Trees
10708, Fall 2020
Pradeep Ravikumar

1 Introduction

Variable Elimination is able to leverage the structure of the graph to reduce the computational complexity of computing marginal probabilities. Suppose we run VE to compute $P(X_1)$: this would entail that X_1 appear last in the elimination ordering. But we are to compute some other marginal e.g. $P(X_2)$ this would then mean that X_2 should appear last in the elimination ordering, so that it needs a different run of VE. This could be expensive. It is instructive therefore to look more closely at the computations performed by VE, and seeing if we can combine these to obtain marginals for all the variables. We first observe that before eliminating a variable X_i, VE form an intermediate factor say ψ_i by multiplying all factors that include X_i: this is an intermediate factors that is a clique in the induced graph corresponding to the VE ordering. Eliminating X_i then creates another factor, say τ_i, that in turn corresponds to a clique in the induced graph. This is then subsequently multiplied together to form an intermediate factor, say ψ_k for eliminating some other variable X_k. We could thus view the VE step as computing a "message" τ_i from one clique factor ψ_i to another ψ_k. VE is thus implicitly working with a data structure involving clique factors and messages between these clique factors. We will now formalize just such a data structure, what is known as a junction tree.

2 Junction Tree

A clique graph for a given UG G is a graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ where the set of nodes $\mathcal{V}$ correspond to cliques in the UG G, each maximal clique is a node in $\mathcal{V}$, and where for any edge $(C_1, C_2) \in \mathbb{E}(\mathcal{H})$, the corresponding cliques have a non-null intersection $C_1 \cup C_2 = $. Each such edge is also associated with a "sep-set" $S_{12} = C_1 \cup C_2$. When this graph has no cycles, and has a single connected component, this is called a clique tree (and a clique forest when there are potentially more than one connected component).

We are interested in clique trees that satisfy a special property.

Definition 1 (Running Intersection Property) *A clique tree $\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ is said to satisfy the* running intersection property *if whenever there is a variable X common to two nodes $C_i, C_j \in \mathcal{V}_\mathcal{T}$, so that $X \in C_i \cup C_j$, then X also occurs in all nodes in the unique path in $\mathcal{T}$ between $C_i$and C_j.*

A clique tree that satisfies the running intersection property is also called a *junction tree*. There is an equivalent characterization of junction trees, in terms of conditional independences.

Proposition 2 *Consider a clique tree $\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathbb{E}_\mathcal{T})$ over an underlying UG G. Then for any edge $(C_i, C_j) \in \mathcal{E}_\mathcal{T}$, let $V_{<(i,j)}$ denote all the variables in cliques in the tree on the C_i side of the edge, and $V_{>(i,j)}$ denote all the variables in cliques in the tree on the C_j side of the edge. Then $\mathcal{T}$ satisfies the* running intersection property *iff for every edge (C_i, C_j) in $\mathcal{E}_\mathcal{T}$, the sep-set $S_{ij} = C_i \cap C_j$ separates $V_{<(i,j)}$ from $V_{>(i,j)}$ in the underlying UG G.*

Proposition 3 *VE over a graph G with factors Φ induces a clique tree with respect to the VE induced graph.*

Proof. Consider a clique graph, where the nodes correspond to the intermediate factors in VE generated right before eliminating a variable. Suppose the residual factor obtained after eliminating a variable from ψ_i is then multiplied to form the factor ψ_j. Then, join ψ_i and ψ_j by an edge. It is clear that: (a) the nodes in this clique tree correspond to maximal cliques in the VE induced graph, and (b) the graph is a tree, since there cannot be a cycle: each intermediate factor corresponds to a variable being eliminated, and these are in a total ordering. □

We can show moreover that this VE clique tree is also a *junction tree*.

Proposition 4 *VE over a graph G with factors Φ induces a junction tree with respect to the VE induced graph.*

Proof. We need to show that the resulting clique tree satisfies the running intersection property. Let C, C' be two clusters that contain a variable X, and let C_X be the cluster formed when X is being eliminated. We will show that X will be present in all clusters in the path between C and C_X, and similarly between C' and C_X, thereby proving the result.

We first note that C_X occurs later in the VE execution than C: since after X is eliminated at C_X, later clusters will not contain X. And in the path from C to C_X, even though other variables are being eliminated, X is not being eliminated, so that X will be part of the messages computed, and hence part of all the clusters in the path from C to C_X. □

3 Sum Product Message Passing

So far we have seen that VE corresponds to passing messages in a clique tree. We now have the machinery to develop an algorithm in the opposite direction. Given a clique tree, and a root, we can perform variable elimination from the leaves to the root.

3.1 Six Node DGM Example

Recall our running example of the six node DGM. Suppose we collate the six node conditional distribution factors into four larger factors (we will see in the sequel a systematic way to do so, for now consider this specific collection of factors): $\psi_1(X_1, X_2, X_3) = P(X_1)P(X_2 \,|\, X_1)P(X_3 \,|\, X_1)$, $\psi_2(X_2, X_3, X_5) = P(X_5 \,|\, X_2)$, and $\psi_3(X_2, X_5, X_6) = P(X_6 \,|\, X_2, X_5)$, and $\psi_4(X_2, X_4) = P(X_4 \,|\, X_2)$. These can then be connected to form a cluster tree, as shown in the following figure.

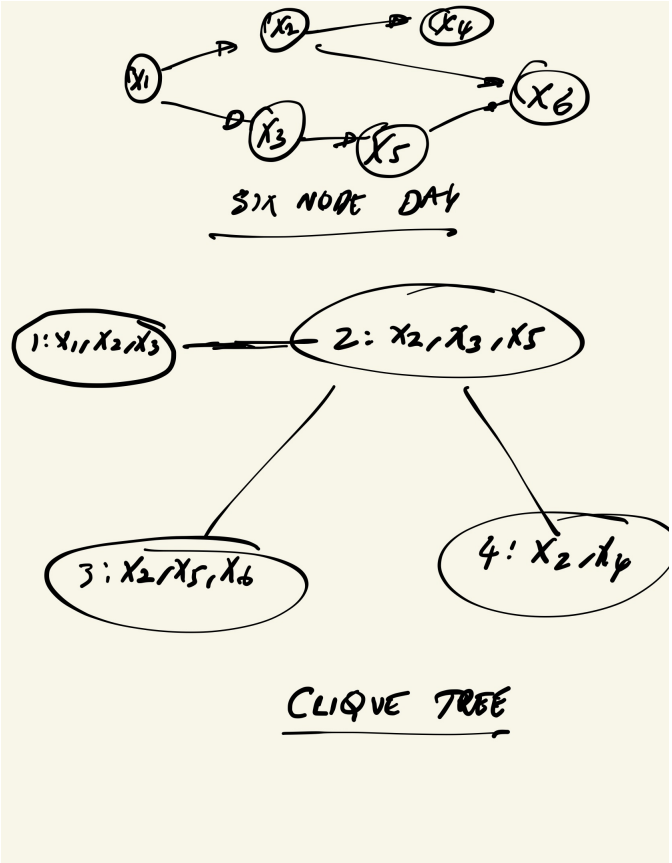


Figure 1: DGM, Clique Tree

Consider a variable elimination ordering: $6, 4, 5, 3, 2, 1$. We can then designate $\psi_1(x_1, x_2, x_3)$ as the root, and then implement VE via messages passed from the leaf factors to the root, as shown in Figure 2. It can then be shown that $\beta_1(X_1, X_2, X_3)$ is actually the marginal over (X_1, X_2, X_3) so that we could in turn compute the marginal over any one of the variables in $\{X_1, X_2, X_3\}$ by marginalizing out the others.

Consider a different variable elimination ordering: $1, 4, 3, 2, 5, 6$. We can then designate $\psi_3(x_2, x_5, x_6)$ as the root, and then implement VE via messages passed from the leaf factors

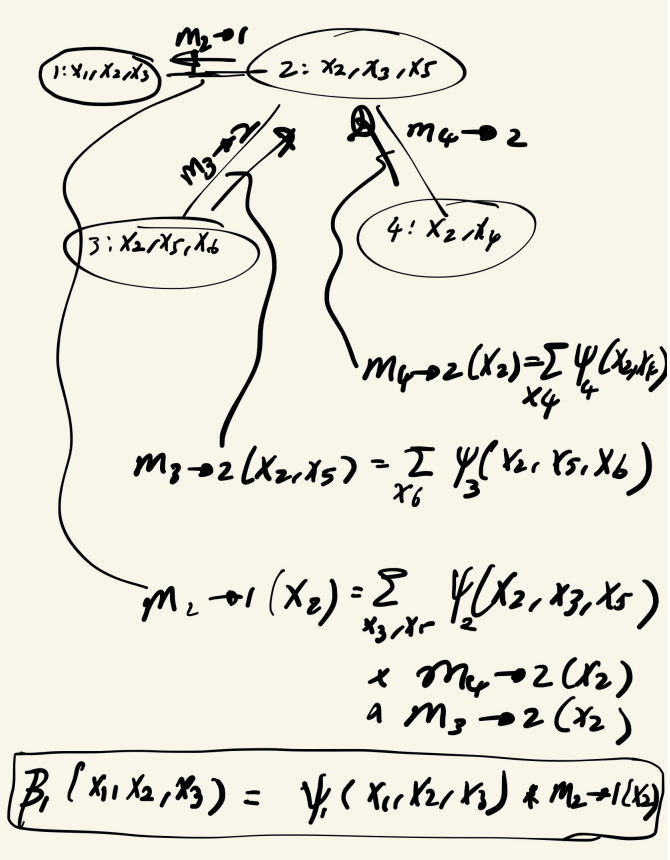


Figure 2: VE as message passing

to the root, as shown in Figure 3. It can then be shown that $\beta_3(X_2, X_5, X_6)$ is actually the marginal over (X_2, X_5, X_6) so that we could in turn compute the marginal over any one of the variables in $\{X_2, X_5, X_6\}$ by marginalizing out the others.

3.2 General Case

Let us now consider the general case. Let T be a junction tree with clique factors $\{C_i\}_{i=1}^k$. Given a set of factors Φ, we assume we assign each factor $\phi \in \Phi$ to some clique factor $\alpha(\phi) \in [k]$ so that we get clique factor functions:

$$\Psi_i(C_i) = \prod_{\phi \,|\, \alpha(\phi)=i} \phi,$$

and where each factor ϕ is assigned to a unique clique factor function so that

$$\prod_{\phi \in \Phi} \phi = \prod_{i=1}^{k} \psi_i.$$

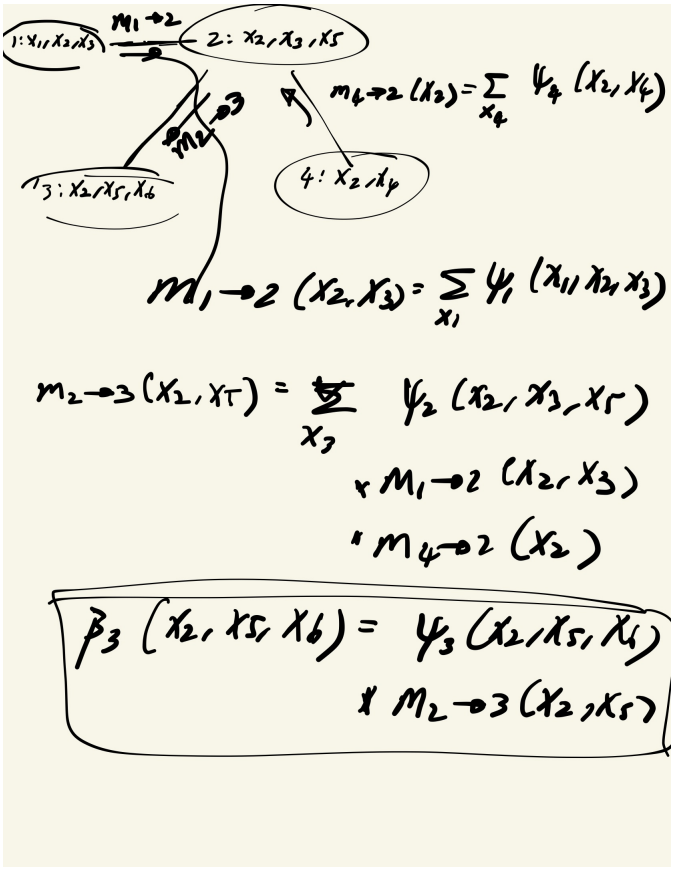


Figure 3: VE as message passing

Sum-Product Message. We can then compute the message from clique factor C_i to clique factor C_j as:

$$m_{i\to j} = \sum_{C_i \setminus C_j} \psi_i \prod_{k \in \text{NBRS}_i - \{j\}} m_{k\to i}.$$

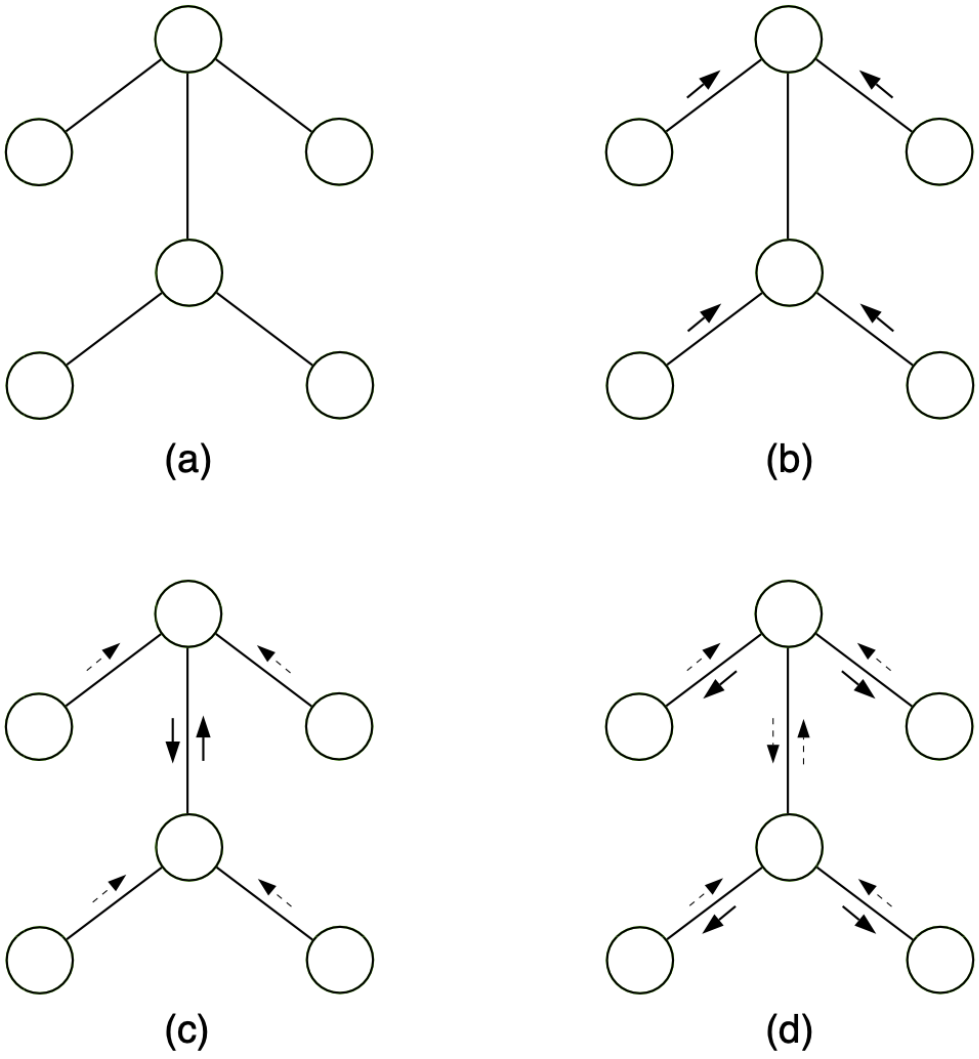
Message Passing Protocol. A clique factor will pass messages to its neighbors while adhering to the following message passing protocol: a clique factor can send a message to a neighboring clique factor when (and only when) it has received messages from all of its remaining neighbors.

There are two common ways to implement this protocol.

The first is to pass all possible messages at any step in parallel. Note that leaf clique factors will always be able to pass messages to their neighbors. And then all nodes one level up from a leaf, and so on. As a prototypical example, consider the clique tree in panel (a) of the figure below. In panel (b), in the first step, leaf factors send messages to their neighbors: this is consistent with the protocol, since they have no other neighbors. In panel(c), the

messages are passed between the backbone factors: this too is consistent with the protocol. Finally in panel (d), messages are passed from the backbone factors back to the leaf factors.

The second way to implement this protocol is to designate a root factor, and then in an upward pass, we first pass messages from the leaf to the root (similar to how we passed the VE messages earlier). And then in a downward pass, pass messages from the root back to the leaf factors.



Beliefs. Once all possible messages are passed, i.e. once in each direction of the edges of the clique tree, then we can compute *beliefs* at each clique factor as:

$$\beta_i(C_i) = \psi_i \prod_{j \in \text{NBRS}_i} m_{j \to i}.$$

We have the following theorem.

Theorem 5 *Given a set of factors Φ over $X = (X_1, \ldots, X_p)$, let $\widetilde{P} = \prod_{\phi \in \Phi} \phi$. Note that $\widetilde{P}(X)$ is equal to the joint density $P(X)$ in the case of DAG factors, and is equal to the unnormalized density in the case of UG factors. Then the messages $\{\beta_i\}_{i \in [m]}$ calculated at the end of message passing in a clique tree with clique factors $\{C_i\}_{i=1}^m$ satisfy:*

$$\beta_i(C_i) = \sum_{\mathcal{X} - C_i} \widetilde{P}(X).$$

Thus, in the case of DGMs, the beliefs exactly correspond to the marginals over the clique factors, while in the case of UGMs, the beliefs are unnormalized marginals. To get the normalized marginals, we simply use the normalized beliefs, so that we have:

$$P(C_i) = \beta_i(C_i) / \sum_{c_i} \beta_i(c_i).$$

The message passing algorithm above is also called the Shafer-Shenoy algorithm.

4 Calibration Message Passing

There is yet another approach to passing messages between clique factors. The Shafer-Shenoy algorithm was derived as a generalization of variable elimination: the message being passed from one clique factor to a neighboring clique factor is as the marginalization of a product factor function (of original clique factor, and other messages).

A simpler alternative is for a clique factor C_i to simply marginalize out from its current belief variables in $C_i \backslash C_j$ and send this marginal over $S_{ij} = C_i \cap C_j$ to the neighboring clique factor C_j. We will be formalizing this alternative message passsing algorithm in the section. We first briefly note that this was developed in parallel by Lauritzen and Spiegelhalter, as well as by Jensen, Olesen and Anderson. It is thus called as the Lauritzen-Spiegelhalter algorithm, as well as the HUGIN algorithm. The latter is because Jensen, Olesen and Anderson used this algorithm in their HUGIN software system. Some also call it the sum-divide algorithm for reasons which will be clear in the sequel.

As before, we assume we have a junction tree T with clique factors $\{C_i\}_{i=1}^k$, and associated clique factor functions ψ_i that consist of products of assigned factors from the given set of factors Φ of the PGM.

Here, we will make use of two sets of belief functions: $\beta_i(C_i)$ for all the clique factors $\{C_i\}_{i=1}^k$, as well as μ_{ij} for the separating sets $S_{ij} = C_i \cup C_j$ for all $(i,j) \in E(T)$.

HUGIN Message. We can then compute the message from clique factor C_i to clique factor C_j as:

$$\begin{aligned} m_{i\to j}^{\text{HUGIN}} &= \sum_{C_i - S_{ij}} \beta_i \\ \beta_j &= \beta_j \frac{m_{i\to j}^{\text{HUGIN}}}{\mu_{ij}} \\ \mu_{ij} &= m_{i\to j}^{\text{HUGIN}}. \end{aligned}$$

Thus, the HUGIN message consists of what clique factor $\psi_i(C_i)$ believes should be the marginal over S_{ij}. We then use this to update both the belief of the recipient clique factor, as well as the belief of the separating set.

The message passing protocol as well as serial and parallel implementations of the protocol are as with the Shafer-Shenoy message passing algorithm.

We have the following theorem.

Theorem 6 *Given a set of factors Φ over $X = (X_1, \ldots, X_p)$, let $\widetilde{P} = \prod_{\phi\in\Phi} \phi$. Note that $\widetilde{P}(X)$ is equal to the joint density $P(X)$ in the case of DAG factors, and is equal to the unnormalized density in the case of UG factors. Then the beliefs $\{\beta_i\}_{i\in[m]}$, and $\{S_{ij}\}_{(i,j)\in E(T)}$ calculated at the end of HUGIN message passing in a clique tree with clique factors $\{C_i\}_{i=1}^m$ satisfy:*

$$\begin{aligned} \beta_i(C_i) &= \sum_{\mathcal{X} - C_i} \widetilde{P}(X) \\ \mu_{ij}(S_{ij}) &= \sum_{\mathcal{X} - S_{ij}} \widetilde{P}(X). \end{aligned}$$

5 Message Passing: Calibration and Reparameterization

Let us now go back to the Shafer Shenoy message passing algorithm. Recall that at the end of the algorithm, the beliefs at each clique factor correspond to the (unnormalized) marginals

over the clique variables. It thus follows that if two clique factors share a set of nodes, then they should agree on the marginal over these shared set of nodes. This local consistency is known as clique calibration.

Definition 7 *Two adjacent cliques C_i and C_j, with beliefs β_i, β_j in a clique tree T are said to be calibrated if*

$$\sum_{C_i - S_{ij}} \beta_i(C_i) = \sum_{C_j - S_{ij}} \beta_j(C_j).$$

We say a clique tree is calibrated if all adjacent cliques are calibrated.

From the discussion in the beginning of the section, it follows that at the end of Shafer-Shenoy message passing, we get a calibrated clique tree. We moreover have the following proposition.

Proposition 8 *At the end of Shafer-Shenoy message passing, we have that:*

$$\beta_i = \psi_i \prod_{j \in \text{NBRS}_i} m_{j \to i} \tag{1}$$

$$\mu_{ij}(S_{ij}) = m_{i \to j}\, m_{j \to i}. \tag{2}$$

Proof. The first statement is just the definition of clique beliefs. For the second, we have that:

$$\begin{aligned}
\mu_{ij}(S_{ij}) &= \sum_{C_i - S_{ij}} \beta_i(C_i) \\
&= \sum_{C_i - S_{ij}} \psi_i \prod_{k \in \text{NBRS}_i} m_{k \to i} \\
&= \sum_{C_i - S_{ij}} \psi_i m_{j \to i} \prod_{k \in \text{NBRS}_i - \{j\}} m_{k \to i} \\
&= m_{j \to i} \sum_{C_i - S_{ij}} \psi_i \prod_{k \in \text{NBRS}_i - \{j\}} m_{k \to i} \\
&= m_{j \to i} m_{i \to j}.
\end{aligned}$$

□

This in turn leads to the following powerful theorem: that message-passing is essentially a reparameterization of the PGM distribution.

Theorem 9 *At the end of Shafer-Shenoy message passing, we have that:*

$$\widetilde{P}_\Phi(X) = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{(i,j) \in E(T)} \mu_{ij}(S_{ij})}.$$

Proof. By the definition of the clique beliefs, the numerator can be written as:

$$\prod_{i \in V_T} \psi_i(C_i) \prod_{k \in \text{NBRS}_i} m_{k \to i}.$$

While using the earlier proposition, the denominator can be written as:

$$\prod_{(i,j) \in E(T)} m_{j \to i} m_{i \to j}.$$

Each $m_{i \to j}$ appears exactly once in the numerator and once in the denominator, so that they cancel and we have that the RHS $= \prod_{i \in V_T} \psi_i = \widetilde{P}_\Phi$. $\square$

Note that even before message passing, we can always write down $\widetilde{P}_\Phi = \frac{\prod_{i \in V_T} \beta_i^{(0)}(C_i)}{\prod_{(i,j) \in E(T)} \mu_{ij}^{(0)}(S_{ij})}$, where $\beta_i^{(0)}(C_i) = \psi_i(C_i)$, and $\mu_{ij}^{(0)}(S_{ij}) = 1$. This is essentially the original PGM factorized form. The message passing algorithm then reparameterizes this same distribution using the true clique and separating set beliefs.

5.1 Equivalence of Shafer-Shenoy and HUGIN

The same properties — calibration, and reparameterization of the unnormalized probability measure — also hold for the HUGIN algorithm.

That the end of HUGIN algorithm results in calibrated beliefs follows simply from its correctness, as with Shafer-Shenoy. To prove that it also results in reparameterization of the unnormalized probability measure, we first show something that would also be of independent interest: there is a step-by-step equivalence between the two algorithms, as elaborated in the following theorem.

Theorem 10 *Consider initializing Shafer-Shenoy algorithm with initial potentials* $\{\Psi_i\}_{i \in V_T}$ *and messages* $\{m_{i \to j}, m_{j \to i}\}_{(i,j) \in E(T)}$*; as well as initializing HUGIN with clique beliefs* $\{\beta_i\}_{i \in V_T}$ *and separating set beliefs* $\{\mu_{ij}\}_{(i,j) \in E(T)}$ *such that the following holds:*

$$\beta_i = \psi_i \prod_{j \in \text{NBRS}_i} m_{j \to i} \tag{3}$$

$$\mu_{ij}(S_{ij}) = m_{i \to j}\, m_{j \to i}. \tag{4}$$

For any pair of neighboring cliques C_i, C_j, suppose we pass messages from C_i and C_j in both of the algorithms. Suppose $\{m'_{i\to j}, m'_{j\to i}\}_{(i,j)\in E(T)}$ are the new set of Shafer-Shenoy messages, and suppose that $\{\beta'_i\}_{i\in V_T}$ and $\{\mu'_{ij}\}_{(i,j)\in E(T)}$ are the new set of HUGIN beliefs. Then (3)*, and* (4) *still hold for the new set of messages and beliefs $m'_{i\to j}, \beta'_j, \mu'_{ij}$.*

As a simple corollary of this we have that:

Corollary 11 *At the end of HUGIN algorithm, we have that:*

$$\widetilde{P}_\Phi(X) = \frac{\prod_{i\in V_T} \beta_i(C_i)}{\prod_{(i,j)\in E(T)} \mu_{ij}(S_{ij})}.$$

Proof. The proof of the corresponding statement for Shafer-Shenoy only used the statements (3), and (4). Since these also hold for HUGIN, the proof follows along similar lines. □

Local to Global Consistency The message passing algorithms can be seen to essentially ensure local consistency of the clique beliefs. But why do these result in global consistency i.e. that the clique beliefs are exactly the unnormalized marginals over the clique variables? The proof for this uses the running intersection property of junctiont trees: without this, the local consistency would not result in global consistency. Intuitively, the RIP ensures a contiguous chain of local consistency for any variable so that all clique beliefs are globally consistent with respect to any marginal.

6 Constructing Junction Trees

Not every UG G has a junction tree: one can always form a tree over cliques in G, but they may not always satisfy the running intersection property. There is however a class of graphs we have considered which admit such junction trees.

Proposition 12 *Every chordal UG G has a junction tree.*

Proof. We can show this by induction. If G is complete then it can be represented vacuously by a junction tree consisting of a single clique containing all the nodes. Otherwise, let X be a simplicial vertex, and let C consist of its neighbors, and $B = V - C - \{X\}$. It can be seen that C separates X from B in G, and if G is chordal, then so is $G[B \cup C]$. By inductive assumption, $G[B \cup C]$ has a junction tree, let us denote this by $\mathcal{T}_B$. Let C' be the maximal clique node in $\mathcal{T}_B$ containing C. We then construct $\mathcal{T}$ by creating a new clique node $C'' = \{X\} \cup C$, and connecting it to C'. For any clique $D \in \mathcal{T}_B$, the intersection

$D \cap C'' = D \cap C \subseteq D \cap C'$. Thus, all variables in $D \cap C'$ also occur in $D \cap C'$, and by assumption all variables in $D \cap C'$ will also occur in all clique nodes in the unique path between D and C' in $\mathcal{T}_B$. Thus the running intersection property is satisfied for $\mathcal{T}$, which is thus a junction tree. $\square$

Given an arbitrary graph G, we generate a junction tree in three steps.

1. Triangulate the graph to get a chordal graph. One way to do that for instance is to run VE (just the graph-theoretic counterpart: contracting variables one by one, and connecting its neighbors, and then finally adding in all fill edges to the original graph). An optimal triangulation that ensures that the size of the largest clique is smallest is NP-hard, but one could use various heuristics (the most scalable are to use the heuristics we discussed earlier for finding a good VE ordering).

2. Given a chordal graph, find its maximal cliques. While finding maximal cliques is NP-hard for general graphs, this can be done simply for chordal graphs. Just find a simplicial vertex: the vertex and its neighbors form a maximal clique. Then eliminate the simplicial vertex, and recurse.

3. Connect the maximal cliques via edges to form a tree. First connect any two maximal cliques that share variables with an edge, with weight equal to the number of shared variables. And find the maximal spanning tree. This will be a junction tree. This follows from the following proposition.

Theorem 13 *A spanning clique tree T is a junction tree iff it is a maximal spanning tree in the weighted graph constructed above.*

Proof. For any clique tree, the following can be seen to hold true:

$$\sum_{(i,j)\in E(T)} \mathbb{I}(X_k \in S_{ij}) \leq \sum_{i \in V_T} \mathbb{I}(X_k \in C_i) - 1,$$

with equality iff it is a junction tree (this is essentially the running intersection property).

We then have that:

$$\begin{aligned}
w(T) &= \sum_{(i,j)\in E(T)} |S_{ij}| \\
&= \sum_{(i,j)\in E(T)} \sum_{k\in[p]} \mathbb{I}(X_k \in S_{ij}) \\
&= \sum_{k\in[p]} \sum_{(i,j)\in E(T)} \mathbb{I}(X_k \in S_{ij}) \\
&\leq \sum_{k\in[p]} \big(\sum_{i\in V_T} \mathbb{I}(X_k \in C_i) - 1 \big) \\
&= \sum_{i\in V_T} \sum_{k\in[p]} \mathbb{I}(X_k \in C_i) - p \\
&= \sum_{i\in V_T} |C_i| - p,
\end{aligned}$$

with equality iff T is a junction tree, since the only inequality is a characterization of the RIP as noted earlier. It thus follows that a spanning clique tree T is a junction tree iff it is a maximum weight spanning tree in the weighted graph constructed earlier. $\square$