# Applying the Junction Tree Algorithm to Variable-Length DBNs

Kevin P. Murphy

5 October 2001

The simplest approach to inference in a DBN is to "unroll" it for $T$ slices and then apply any inference algorithm to the resulting static Bayes net. (By inference, we mean offline smoothing, i.e., computing $P(F_i|O_{1:T})$ for all families $F_i$.) Since we are interested in computing the marginals of all families simultaneously, we use the junction tree algorithm, which uses dynamic programming to avoid the redundant computation that would be involved in calling variable elimination once per family.

If sequences can have variable lengths, it becomes too expensive to repeatedly unroll and triangulate the DBN. One approach is to unroll once, to some maximum length $T_{\max}$, and then to "splice out" redundant cliques from the junction tree when doing inference on a shorter sequence [Zwe97]. To ensure there is some repetitive structure to the junction tree which can be spliced out, we must use a constrained elimination ordering, in which we eliminate all nodes from slice $t$ before any from slice $t+1$. The temporal constraint ensures that we create "vertical" cliques, which only contain nodes from neighboring time-slices, instead of "horizontal" cliques, which can span many time-slices. The resulting jtree is said to be constrainedly triangulated.

An alternative to splicing out is to explicitly identify the repetitive structure, and repeat it when necessary. This approach can then be used for online inference, and avoids the need to specify $T_{\max}$ and to store a large junction tree. An algorithm to do this can be found in [Kja95], but is quite complicated. We now present a significant simplification.

We start by defining the interface of a time-slice to be all the nodes that have incoming temporal arcs, or are parents of nodes with incoming temporal arcs. (The resulting set of nodes will be a separator in the

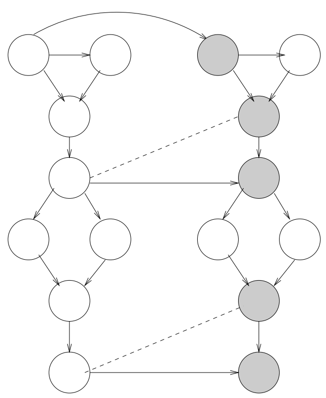

Figure 1: A DBN from [Kja95]. The shaded nodes are the interface nodes in slice 2. The dotted undirected arcs are moralization arcs.

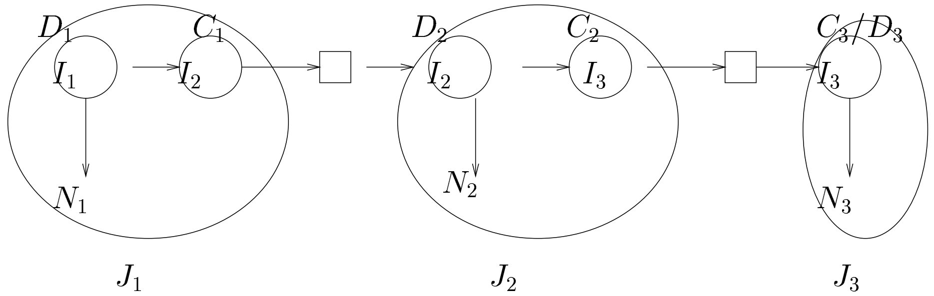


Figure 2: A schematic illustration of how to join the junction trees for each $1\frac{1}{2}$-slice DBN. $I_t$ are the interface nodes for slice $t$, $N_t$ are the non-interface nodes. $C_t$ is the clique in $J_t$ containing $I_{t+1}$, $D_t$ is the clique in $J_t$ containing $I_t$. The square box represents a separator, whose domain is $I_t$.

moral graph: see Figure 1 for an example.) We then define the "$1\frac{1}{2}$-slice DBN" to be the DAG created by eliminating all non-interface nodes (and their arcs) from the 2-slice DBN starting at time $t$. Kjaerulff calls this $\mathcal{G}_t$, but we will call it $H_t$ ($H$ for half). This contains nodes $I_t$, $N_t$ and $I_{t+1}$, where $I_t$ are the interface nodes in slice $t$ and $N_t$ are the non-interface nodes.[1] Kjaerulff [Kja95] proves that if we constrainedly triangulate each $H_t$ separately, and then join up each triangulated graph (by taking the union of the nodes and edges), then the resulting compound graph, $H^*$, will also be constrainedly triangulated. Furthermore, each interface will be a complete separator in $H^*$. In fact, it is not necessary to use a constrained elimination, as long as we ensure that each junction tree contains cliques with $I_t$ and $I_{t+1}$, since each interface will still separate the past from the future.

What this means is that we can "glue" all the junction trees together via their interfaces, as shown in Figure 2. We can perform inference in each tree by calling collect-evidence and then distribute-evidence (see e.g., [CDLS99, HD96] for details of these procedures). The question is: how do we combine the information from each tree? Let $J_t$ be the junction tree for $H_t$, $C_t$ be a clique-node that contains $I_{t+1}$ in $J_t$ (the out-interface), $D_t$ be a clique-node that contains $I_t$ in $J_t$ (the in-interface), $\phi_C$ be the potential associated with $C$, and similarly for $\phi_D$. We can pass information from $J_{t-1}$ to $J_t$ by performing an absorption

$$\phi_D^* = \phi_D \times \frac{\sum_{C \setminus D} \phi_C}{\sum_{D \setminus C} \phi_D}$$

where $C = C_{t-1}$ and $D = D_t$. If we make $C_t$ the root of each $J_t$, we can do a collect-evidence operation, pass this information on to $D_{t+1}$, and then repeat. Once we reach the final slice, we can distribute the evidence from $C_t$ (to make all the potentials within $J_t$ consistent), compute the potential on $D_t$, pass it on to $C_{t-1}$, and repeat. We give the details below.

## 0.1 Forwards pass

In [Mur99], we suggest delaying the initialization of the potentials in a junction tree until the evidence arrives. We can go further, and delay their initialization until any priors arrive. Hence the potentials in $J_t$ will be all 1's initially, so we can simplify the absorption operation in the forwards pass to $\phi_D^* = \phi_D \times \sum_{C \setminus D} \phi_C$. Similarly, we can multiply on other sources of information onto the potentials of $J_t$, an operation we denote by $\phi_C(J_t) *= \phi$, where $C$ is some clique-node in $J_t$, and $\phi$ is some potential whose domain is a subset of $C$'s. For example, $\phi$ might be the CPD for some node in slice $t$.

Before we give the pseudo-code for the algorithm, we need some more notation. Let $P_t$ be the persistent

[1] For $t = T$, we define $H_T$ to just contain the single slice containing $I_T$ and $N_T$.

nodes in slice $t$ (i.e., nodes which have incoming temporal arcs), and $T_t$ be all the rest (i.e., the transient nodes). (Note that $P_t \subseteq I_t$.) Let $\text{clq}(J_t, S)$ be the clique-nodes in $J_t$ containing nodes $S$, $\text{CPD}(S)$ be the potentials containing the CPDs for nodes $S$, and $\text{dom}(C)$ be the domain of clique-node $C$.

The main steps of the inner-loop of the forwards-pass (for $t = 2, \ldots, T-1$) are as follows:

1. Let $C_t = \text{clq}(J_t, I_{t+1})$ and $D_t = \text{clq}(J_t, I_t)$.
2. Initialise $\phi(J_t)$ to all 1's.
3. $\phi_{D_t}(J_t) *= \hat{P}_{t|t-1}$.
4. $\phi_c(J_t) *= \text{CPD}(S)$ where $S = T_t \cup P_{t+1}$ and $c = \text{clq}(J_t, S)$.
5. $\phi(J_t) =$ collect-evidence to $C_t$
6. $\hat{P}_{t+1|t} = \sum_{\text{dom}(C_t) \setminus I_{t+1}} \phi_{C_t}$

$\hat{P}_{t|t-1} \stackrel{\text{def}}{=} P(P_t | O_{1:t-1})$ is the one step-ahead prediction of the persisent nodes. We don't include the CPDs for $P_t$ in $J_t$ since they were already included in this prediction term from $J_{t-1}$.

The steps for $t = 1$ are basically the same, except that instead of including $\hat{P}_{1|0}$, we include the prior CPDs for the persistent nodes $P_1$. For $t = T$, we don't include $P_{t+1}$ and don't compute $\hat{P}_{T+1|T}$.

## 0.2 Backwards pass

The backwards pass is even simpler.

1. Let $C_t = \text{clq}(J_t, I_{t+1})$, $D_t = \text{clq}(J_t, I_t)$ and $C_{t-1} = \text{clq}(J_{t-1}, I_t)$.
2. $\phi(J_t) =$ distribute-evidence from $C_t$
3. Set
$$\phi_C^* = \phi_C \times \frac{\sum_{D \setminus C} \phi_D}{\sum_{C \setminus D} \phi_C}$$
   where $C = C_{t-1}$ and $D = D_t$.

For $t = 1$, we simply omit the final absorption step.

## 0.3 Normalization

As with HMMs, it is essential to normalize the probabilities at each step to prevent numerical underflow. We do this by normalizing all the potentials after the collect/distribute steps. To compute the log-likelihood, we must keep track of the normalization constants. We explain how to do this in the case of HMMs; the same method can be easily be extended to the general DBN case.

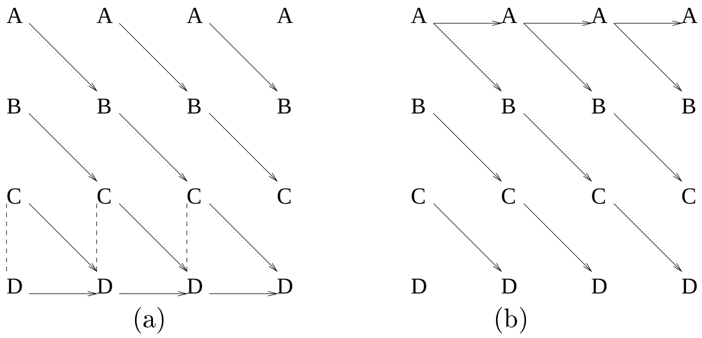


Figure 3: Two unrolled DBNs. (a) The nodes in the last slice do not become connected when we eliminate the previous slices. (Dotted lines represent moralization arcs.) (b) The nodes in the last slice *do* become connected when we eliminate the first 3 or more slices.

Define the normalized prior as $\alpha_{t-1}(i) \stackrel{\text{def}}{=} P(Q_{t-1} = i|O_{1:t-1})$. We can compute the normalized posterior as follows.

$$\begin{aligned}
\alpha_{t|t-1}(j) &\stackrel{\text{def}}{=} P(Q_t = j|O_{1:t-1}) = \sum_i P(Q_t = j|Q_{t-1} = i)P(Q_{t-1} = i|O_{1:t-1}) \\
\hat{\alpha}_t(j) &\stackrel{\text{def}}{=} P(O_t, Q_t = j|O_{1:t-1}) = P(O_t|Q_t = j)\alpha_{t|t-1}(j) \\
c_t &\stackrel{\text{def}}{=} P(O_t|O_{1:t-1}) = \sum_j \hat{\alpha}_t(j) \\
\alpha_t(j) &= \hat{\alpha}_t(j)/c_t
\end{aligned}$$

The log-likelihood is given by

$$L = \log\left(P(O_1)P(O_2|O_1)P(O_3|O_{1:2})\ldots\right) = \log \prod_{t=1}^{T} c_t = \sum_{t=1}^{T} \log c_t$$

## 0.4 Can we do better?

The above algorithm requires that the junction tree contain at least two cliques, one of which contains $I_t$ and one of which contains $I_{t+1}$. A natural question is: do we have to create such large cliques? The following theorem suggests that we do.

**Theorem** [[RTL76], see also [Jen01, p.187]]. Let $A_1, \ldots, A_n$ be an elimination sequence triangulating the (moral) graph $G$, and let $A_i$, $A_j$ be two non-neighbors in $G$, $i < j$. Then the elimination sequence introduces the fill-in $A_i - A_j$ iff there is a path $A_i - X_1 - \ldots - A_j$ such that all intermediate nodes $X_k$ are eliminated before $A_i$.

Consider the example in Figure 1, and suppose we eliminate all nodes in slice 1 before any in slice 2. Because the moral graph of slice 1 is one connected component, and all interface nodes in slice 2 have a parent in slice 1 (by definition), they will be completely connected in the triangulated graph, and hence $I_2$ will appear in a clique. If you only process two slices, you might be able to do better using a different (unconstrained) elimination sequence. However, in general the problem cannot be avoided. For instance, if we unroll $T$ slices, and eliminate the first $T-1$ in an unconstrained fashion, and then eliminate slice $T$, the interface nodes in the last slice will become connected.

Not all examples are like this. In Figure 3(a), we show a DBN where the interface contains all the nodes in the slice. However, the largest clique in both the constrained and unconstrained jtree is the triple $\{C_t, D_t, C_{t+1}\}$. In Figure 3(b), we have made a slight change to the model; now it is clear that if we eliminate the first 3 slices, then slice 4 becomes a complete set. Indeed we find that the largest clique in the constrained jtree is $\{A_{t-1}, A_t, B_t, C_t, D_t\}$. In the unconstrained jtree, however, each clique is just a pair of nodes.

Clearly a constrained elimination ordering cannot do any better than an unconstrained elimination ordering (better in terms of minimizing the maximum clique size). However, by using a constrained ordering, we can identify repetitive structure, which allows us to handle variable-length sequences more efficiently, and enables online inference. In Figure 3(a) we saw an example where the interface nodes were not fully connected even in the constrained junction tree, but such cases are extremely rare.[2] Hence in practice we doubt that it is possible to improve on the above algorithm.

## References


[CDLS99] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer, 1999.

[HD96] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *Intl. J. Approx. Reasoning*, 15(3):225–263, 1996.

[Jen01] F. V. Jensen. *Bayesian Networks and Decision Graphs.* Springer-Verlag, 2001.

[Kja95] U. Kjaerulff. dHugin: A computational system for dynamic time-sliced Bayesian networks. *Intl. J. of Forecasting*, 11:89–111, 1995.

[Mur99] K. P. Murphy. A variational approximation for Bayesian networks with discrete and continuous latent variables. In *UAI*, 1999.

[RTL76] D. Rose, R. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. on Computing*, 5:266–283, 1976.

[Zwe97] G. Zweig. *Speech Recognition with Dynamic Bayesian Networks.* PhD thesis, U.C. Berkeley, Dept. Comp. Sci., 1997.


[2] Another obvious example when the interface nodes do not have to be in the same clique is when we have fully independent chains. It is not clear how to characterize such “degenerate” DBNs, however.