The Efficient Propagation of Arbitrary Subsets of Beliefs in Discrete-Valued Bayesian Belief Networks

Duncan Smith
Department of Physics, Astronomy and Mathematics
University of Central Lancashire
England
Scotland
PR1 2HE

Abstract

The paper describes an approach for propagating arbitrary subsets of beliefs in Bayesian Belief Networks. The method is based on a multiple message passing scheme in junction trees. A hybrid tree structure is introduced, both for the propagation of evidence and as an efficiently permutable representation of a decomposable graph. The use of maximal prime subgraph decompositions and tree permutations to reduce computational cost is demonstrated.

1 INTRODUCTION

A discrete valued Bayesian Belief Network (BBN) is a sparse representation of a full joint probability distribution over a set of variables and is generally represented as a Directed Acyclic Graph (DAG). It is a collection of conditional and marginal probability tables such that each child node is conditional upon its parent nodes. The sparseness of the representation allows evidence to be propagated through the network to efficiently generate posterior beliefs on queried variables. This is achieved through a process of node elimination. The process of node elimination is managed somewhat differently by junction tree methods (Lauritzen and Spiegelhalter 1988) (Shafer and Shenoy 1990) and factoring methods (Li and D'Ambrosio 1994). The former are generally used to propagate the posterior joints for each clique, and subsequently all single beliefs. Factoring methods are generally used to generate one single or joint posterior belief, although they can also be used to propagate all single beliefs (Bloemeke and Valtorta 1998).

A DAG, D, can be converted to an undirected graph, $G=(V, E)$, by pairwise connecting all parental sets and dropping the directions on the edges, such that adjacency between two variables v, w in the G, denoted $v \in \mathrm{Adj}(w)$, implies that there exists some probability table that contains both v and w. Thus initially each table is a pairwise-connected subgraph in G, and is said to be *complete*. The elimination of a variable v implies the deletion of v and all its incident edges from G and the addition of the *fill-in* edges required to make $\mathrm{Adj}(v)$ complete. The set of fill-in edges implied by a node elimination is its deficiency. The set of fill-in edges implied by an elimination ordering, α, is a graph *triangulation* T, such that the corresponding *decomposable* graph is $G'=(V,E\cup T)$. A triangulation is minimal if there exists no $T'\subset T$ such that $G''=(V,E\cup T')$ is triangulated. A complete subgraph of G is a clique C, if there exists no superset of C that is also complete. A separator S, of two complete subgraphs is the intersection of their vertex sets.

As both junction tree and factoring methods are based on a process of variable elimination, they both have computational costs that depend heavily on the chosen elimination ordering. The optimisation of junction trees is usually carried out as an off-line process. Factoring methods exploit Markov properties to reduce the cost of propagation through, for instance, graph pruning (Zhang and Poole 1994). Thus the optimisation problem is query-dependent and is usually carried out as an on-line process. Single joint queries are easily handled by factoring methods whereas the standard junction tree method is to fire additional messages through the junction tree (Jensen 1996). An alternative approach by Xu (1995) creates new cliques which are supersets of existing cliques such that each superset contains all the variables in a queried joint.

2. OPTIMAL DECOMPOSITIONS

Previous attempts to optimise inference in belief networks have used a range of optimisation criteria. The optimisation criterion used here is that of the 'marginalisation cost'. That is, the total cost of generating the queried posteriors in terms of the total number of additions. This cost is essentially equal to the sum of clique state spaces minus the sum of separator state spaces. The use of this criterion, rather than the more usual sum of clique state spaces, is a consequence of a general 2-stage approach to optimisation that attempts to minimise the total cost of additions, before using heuristics to reduce the total cost of table combinations. Strategies for table combination will not be detailed in this paper.

Any non-minimal graph triangulation can be made minimal by removing superfluous edges, one at a time, such that each visited graph is decomposable. This result has been used to thin superfluous edges in order to find minimal triangulations from some arbitrarily triangulated graph (Kjaerulff 1990). Kjaerulff showed that each edge removal results in the breaking up of some clique into two smaller cliques. It is easily shown that the sum of state spaces of any two cliques has, at most, the same state space as a clique formed by their union. As the marginalisation cost requires the subtraction of the separator state space, any superfluous edge removal will reduce the marginalisation cost of a graph triangulation, and any minimum cost solution will have a minimal graph triangulation. Furthermore, in a graph *G*, there exists a subset of clique separators that are common to all minimal triangulations (Leimer 1993). These separators decompose a graph into unique maximal prime subgraphs (MPS) that can be (minimally) triangulated independently of each other.

3. BELIEF PROPAGATION

3.1. INDIVIDUAL JOINT BELIEFS

The undirected graph for the ‘Asia’ network is shown in Figure 1. It has a single fill-in edge {D,E}. The corresponding junction tree is shown in Figure 2, with the separators of the maximal prime subgraphs signified by the more heavily lined boxes.

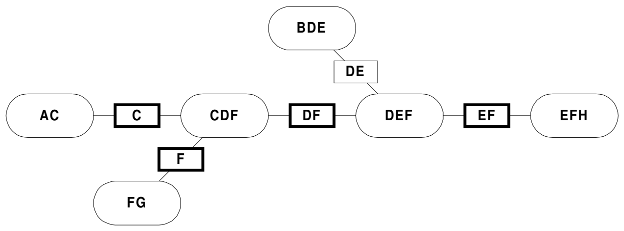

Figure 1: Junction Tree For Asia

Variable firing (Jensen 1996) generates the joint P(A,B) by treating each level of A as observed, and ‘firing’ a message from clique (A,C) to clique (B,D,E). The marginalisation cost of this scheme can be easily evaluated by adding A to all cliques and separators on the path between (A,C) and (B,D,E). This results in the junction tree shown in Figure 3, where A* denotes that A

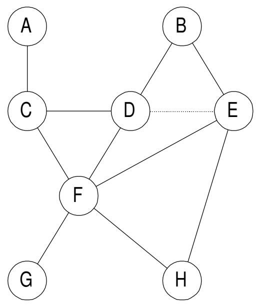

Figure 2: Triangulation For Asia

is a member of a clique due to the query. So in effect, the joint is found by choosing not to eliminate A from cliques containing A*. The new junction tree implies the triangulation in Figure 4, which represents a relatively arbitrary graph decomposition with edges fanning out from A to all variables contained in cliques from (A,C) to (B,D,E). The only constraint on the decomposition is that the variables in the queried joint should form a complete subgraph. So if the undirected graph is pre-processed by making the queried joint complete, then the resulting *query graph* can be re-triangulated in order to reduce the cost of inference.

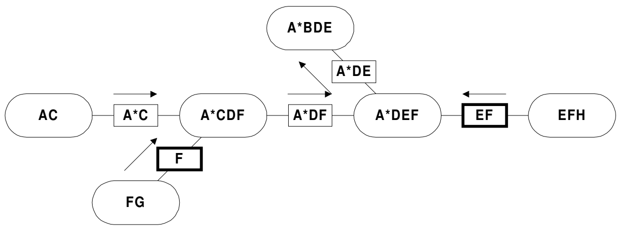

Figure 3: Junction Tree For P(A,B)

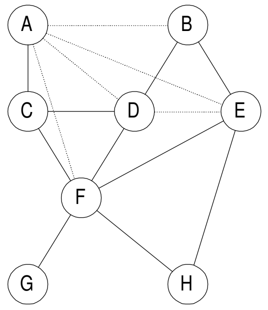

Figure 4: Triangulation For P(A,B)

The additional edges in the query graph can affect the maximal prime subgraph decomposition. In this example the MPSs (A,C), (C,D,F) and (B,D,E,F) are merged to form the MPS (A,B,C,D,E,F) in the query graph. But as a minimal solution is being sought, any existing solutions to other MPSs can be retained, and the problem becomes that of re-triangulating the MPS (A,B,C,D,E,F) in the query graph.

3.2. BELIEFS IN DISTINCT MAXIMAL PRIME SUBGRAPHS

Consider the query set P(A,B), P(H) and the query graph for P(A,B). Generation of P(H) requires additional messages from (A*,B,D,E) to (A*,D,E,F) and from (A*,D,E,F) to (E,F,H). These messages do not require the starred variables. The message from (A*,B,D,E) to (A*,D,E,F) is generated at a cost of bde – de (where lower case denotes a variable's state space). The message from (A*,D,E,F) to (E,F,H) can be computed in a number of ways. It requires the combination of the incoming messages (A*,D,F), (D,E) and any tables allocated to (D,E,F) in the original junction tree, and elimination of A* and D. As A is starred it can be eliminated before any table combinations. So the cost of the message is adf – df for the elimination of A plus the cost def – ef for the elimination of D after table combinations. So,

1. Starred variables are passed in messages in one 'direction' only. Messages in the reverse direction ignore starred variables.
2. When a message requires the elimination of starred variables from a clique, they are eliminated from the clique's existing incoming messages before table combination.

If the elements of the query set are in distinct maximal prime subgraphs in the query graph, then they can be considered individually. The required incoming and outgoing messages to/from an MPS are known from the query and all the costs of MPS solutions are mutually independent.

3.3. MULTIPLE BELIEFS IN THE SAME MAXIMAL PRIME SUBGRAPH

Elements of a query in the same MPS cannot be considered independently. They could be answered separately by producing separate solutions, but this would waste opportunities for re-using results. A naïve extension of the query graph approach for an individual joint to the more general problem of multiple joints might seem inappropriate. In the worst case the graph would be fully connected. But a considered treatment of the starred variables shows that it is possible to generate multiple joints efficiently despite the apparent density of the graph triangulation.

Consider the query set P(A,B), P(H), P(E,G) and the resulting junction tree in Figure 5. All the required messages have been shown with arrows. The message passing that relates directly to the query set is detailed below.

P(A,B) is answered as before by propagating messages from pendant cliques towards (A*,B,D,E) and then marginalising to P(A,B). But when the outgoing message from (A*,D,E,F,G*) is calculated Rule 2 is invoked. G* is eliminated from the incoming message before being combined with incoming message (A*,D,F) and any tables allocated to (A*,D,E,F,G*). F is eliminated from the resultant table to produce the message to be sent to (A*,B,D,E).

P(H) is found by marginalisation from the clique (E,F,H). The message from (A*,D,E,F,G*) is calculated by combining the tables allocated to (A*,D,E,F,G*) with it's incoming messages from (F,G*), (A*,D,F) and (D,E) after eliminating G* and A* under Rule 2, followed by the elimination of D. Note: Only the message (D,E) is sent from (A*,B,D,E) under Rule 1.

The calculation of P(E,G) involves a permutation of the junction tree in order to avoid adding G* to (A*,C,D,F) and (A*,D,F). As usual, Rule 2 is invoked and A* is eliminated from the message (A*,D,F). The resultant table is combined with the other inmessages to (A*,D,E,F,G*), including (F,G*), and P(E,G) is marginalised from the resulting joint, P(D,E,F,G).

So, although ostensibly there is a large clique (A*,D,E,F,G*), its full joint is never calculated. The queried joints are only marginalised from the joints that would be necessary under a single-joint variable firing scheme. Under the proposed scheme cached results are re-used and only the messages necessary for the query set are sent. The optimisation of the re-use of results from intermediate table combinations and eliminations of starred variables is treated as a secondary problem to that of minimising the marginalisation cost. Each clique constitutes a separate combination problem. However, the approach to managing the table combinations will not be detailed here. It is partially managed within a hybrid junction tree structure which can also be efficiently permuted in order to introduce an element of on-line optimisation.

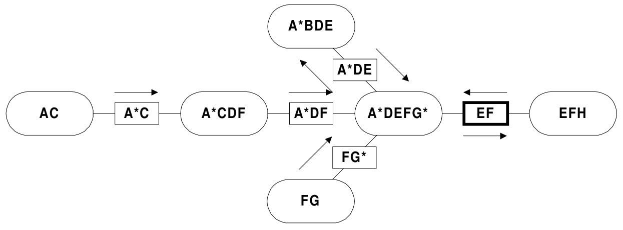

Figure 5: Junction Tree For P(A,B), P(H), P(E,G)

4. DIRECTED JUNCTION TREES

A directed junction tree (DJT) of a connected graph G=(V,E) represents the decomposition G'=(V,E∪T) implied by an elimination ordering α of the vertices of G. Each vertex in G has a corresponding labeled node in a directed junction tree J, of G. A node label Σv indicates the elimination of v from the elimination graph G_V of the set of vertices V s.t. $x \in V$ if and only if there is a directed path $\Sigma x \rightarrow \Sigma v$ in J. Each node Σv contains the member set $v \cup$Madj(v) where x∈Madj(v) denotes that $x \in$ Adj(v) in G_V. Furthermore, there exists a directed edge from Σv to Σx, denoted Σx=Madj(Σv), only if x∈Madj(v). Thus a DJT is essentially a hybrid junction tree / factor tree but with the following additional property.

The directed junction tree property:

$$\Sigma x = \text{Madj}(\Sigma v) \Rightarrow x \in \text{Adj}(v) \text{ in } G' = (V, E \cup T)$$

A DJT has a root node corresponding to the last variable in α. The root node can be chosen arbitrarily as it places no constraints on the underlying graph decomposition.

4.1. PERMUTATION OF DIRECTED JUNCTION TREES

The directed junction tree property enables a DJT to be efficiently permuted in order to search the space of underlying graph triangulations. Permutation of the DJT is achieved by selecting two nodes Σv and Σw=Madj(v), reversing the direction of their common edge and locally updating their members and adjacent edges. Each permutation involves the local re-triangulation of the subgraph induced by the members of the permuted nodes. Care is taken to preserve the directed junction tree property:

1. All edges between Σx_i and Σw s.t. $\Sigma x_i \rightarrow \Sigma w$ are unchanged.
2. Edges between Σx_i and Σv s.t. $\Sigma x_i \rightarrow \Sigma v$ and $w \notin x_i \cup$Madj(x_i) are unchanged.
3. Edges between Σx_i and Σv s.t. $\Sigma x_i \rightarrow \Sigma v$ and $w \in x_i \cup$Madj(x_i) are removed and edges $\Sigma x_i \rightarrow \Sigma w$ are added.

The directed junction tree property guarantees that the two variables eliminated are adjacent in G. Permuting variables that are adjacent in some elimination ordering, but not adjacent in G, leaves the underlying graph decomposition unchanged.

The marginalisation cost can be updated locally as the cost change depends only on the domain sizes of the members of Σv and Σw. It should be noted that although the sum of DJT node state spaces is greater than that of the sum of clique state spaces of an equivalent junction tree, their marginalisation costs are identical.

Lemma. *The space of all graph triangulations corresponding to elimination orderings can be traversed through permutations of a DJT.*

Proof: Any elimination ordering is trivially reachable from any starting ordering by the repeated permutation of adjacent variables. So it is sufficient to show that all corresponding DJT permutations are possible. If two adjacent variables v and w s.t. v is ordered before w, are separated in G, then the pre- and post-permutation orderings have the same DJT. If v and w are adjacent in both the ordering and G, then Σw=Madj(Σv) and there is a DJT permutation corresponding to the reordering of the variables.

Figure 6 shows a sub-DJT corresponding to the MPS (A,B,C,D,E,F) in the decomposition of the Asia network in Figure 2. In this case the root node contains the variables of the MPS separator (E,F). There is no node ΣF as this would allow permutations resulting in a variable on the MPS separator being eliminated before elimination of all non-separator variables. This would lead to non-minimality.

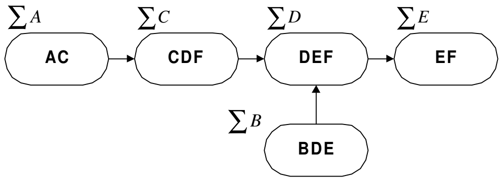

Figure 6: Sub-DJT For The MPS (A,B,C,D,E,F)

The addition of starred variables for the query P(A,B) results in the tree in Figure 7. However this is not a valid DJT. In order to construct a valid DJT, permutations of the DJT in Figure 6 are carried out until the query edge {A,B} appears in a DJT node. This is easily achieved by permuting so that A and B are eliminated late in the ordering.

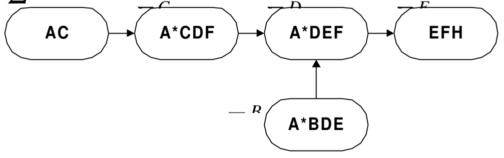

Figure 7: P(A,B) Removes The DJT Property

The tree resulting from the permutations A,C,B,D,E →C,A,B,D,E→C,A,D,B,E→C,D,A,B,E results in the tree in Figure 8. Although the edge {A,B} is a fill-in edge, it can subsequently be treated as a graph edge, ensuring that it will exist in some node after further permutations.

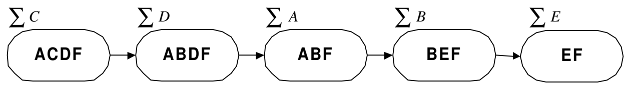

Figure 8: DJT Property Restored By Permutation

Local cost updating must take into account the fact that messages might not be transmitted in both directions between two adjacent cliques. Furthermore, message passing from a clique might be achieved at a lesser cost than the clique's state space minus the message state space. The former can be adequately allowed for by suitably weighting the relevant cliques and separators. The latter is less straightforward and is handled by an alternative DJT permutation scheme.

The DJT for the query P(A,B) could have been permuted by treating the state spaces of the starred variables as weights and ignoring them for the purposes of DJT permutation. In such a case it would be unnecessary to permute the DJT until the edge {A,B} existed in some DJT node. However, it would be necessary to consider the problem of which (possibly intermediate) clique should receive the fired messages.

DJT algorithms and permutation strategies are the subject of on-going research.

4.2. BELIEF PROPAGATION IN DIRECTED JUNCTION TREES

It is easy to construct a junction tree from a DJT by merging adjacent nodes to form cliques, and dropping the directions on the edges. However this is unnecessary as evidence propagation can be handled within the DJT itself. In order to reduce storage costs the DJT is modified as follows.

1. Each probability table is allocated to a node according to some heuristic.
2. Any node which is not a clique, has only a single *parent*, and no allocated table, is merged with its parent.
3. The directions on edges are dropped.

The resulting tree of complete subgraphs is not generally a tree of cliques. The subsequent message passing scheme will not be detailed here as it is essentially a version of the well-known Shafer-Shenoy algorithm (Shafer and Shenoy 1990). For arbitrary sets of beliefs it also obeys the rules outlined earlier.

5. CONCLUSIONS

Existing methods for propagating joint beliefs in junction trees take an existing junction tree and, either implicitly or explicitly, create new cliques which are supersets of existing cliques. The costs of these schemes are arbitrarily high. Factoring methods can easily answer single joint queries, but do not offer a means of propagating multiple joint beliefs without constructing separate factorings. However, it is possible to efficiently generate an arbitrary set of posterior beliefs through a multiple message passing scheme which exploits marginalisation from existing messages, as well as from cliques. Moreover, a directed junction tree representation enables a junction tree to be efficiently permuted, so that elements of an efficient off-line solution can be combined with on-line optimisation. In fact, it is possible to abandon the standard junction tree representation and propagate beliefs in a directed junction tree.

Directed junction trees offer an efficient means of searching the space of decomposable graphs. A number of algorithms have already been implemented, and an adaptation of the scheme for (decomposable) model determination is the subject of current research.

References

Bloemeke, M. & Valtorta, M. (1998). A Hybrid Algorithm to Compute Marginal and Joint Beliefs in Bayesian Networks and Its Complexity. In G.F. Cooper, M. Serafín (eds.), *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp.16-23. San Francisco, Calif: Morgan Kaufmann.

Jensen, F.V. (1996) *An Introduction to Bayesian Networks*. UCL Press, London. pp.60-61.

Kjaerulff, U. (1990). Triangulation of Graphs-Algorithms Giving Small Total State Space. *Technical Report* R90-09. Department of Mathematics and Computer Science, Aalborg University, Denmark.

Lauritzen, S.L. & Spiegelhalter, D.J. (1988). Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. *Journal of the Royal Statistical Society* B, 50, No.2., pp.157-224.

Leimer, H.-G. (1993). Optimal Decomposition by Clique Separators. *Discrete Mathematics* **113**. pp.99-123.

Li, Z. & D'Ambrosio, B. (1994). Efficient Inference in Bayes Networks as a Combinatorial Optimization Problem. *International Journal of Approximate Reasoning* **11**. pp.55-81.

Shafer, G. & Shenoy, P. (1990). Probability propagation. *Annals of Mathematics and artificial Intelligence* **2**. pp.327-352.

Xu, H. (1995). Computing Marginals for Arbitrary Subsets from Marginal Representation in Markov Trees. *Artificial Intelligence* **74**. pp.177-189.

Zhang, N.L. & Poole, D. (1994). A Simple Approach to Bayesian Network Computations. *Proceedings of the 10th Biennial Canadian Artificial Intelligence Conference (AI-94)*, *Banff*, pp.171-178.