A Best-First Search Method for Anytime Evaluation of Belief Networks

N. Jitnah and **A. E. Nicholson**
Department of Computer Science
Monash University
Clayton, VIC 3168, Australia
njitnah,annn@cs.monash.edu.au

Abstract

We present a method for incremental evaluation of a Belief Network (BN). Evaluation is initially performed on a restricted number of nodes in the immediate vicinity of the query nodes. The BN is then traversed radially out from each query node and estimates for the belief of the query node are computed iteratively. This incremental evaluation results in a form of anytime algorithm. A *best-first* graph traversal strategy requires an assessment of the relative importance of various nodes in terms of contributing the most towards a query node. At each step, we must visit in priority the most significant nodes while making a trade-off with computation cost. We use the concept of *arc weights* in a BN to determine to what extent a node influences the query node. We also incorporate a measure of the *computation cost* of visiting a node, in terms of the state space sizes of the node and of its parents.

Basic Framework

BN structure

Given a BN with a set of designated query nodes $\mathbf{Q}$, we iteratively compute estimates for the belief of each $Q \subset \mathbf{Q}$ by traversing the network radially out from Q. For every node or group of nodes visited, the belief of Q is updated by recomputing the relevant term. Thus, each recomputation takes into account the influence of a new node or group of nodes on the query node Q. In the limit, this algorithm returns the exact belief when all the nodes of our BN are visited. However, if we have a very large BN, or possibly one with infinitely many nodes, we can stop once the computed estimate is adequate or when computing resources have run out. We should therefore concentrate on the task of visiting the *best* nodes at each recomputation.

Assume our BN has no underlying loops. In addition to the CPDs from the original BN, each node N is assigned two real-valued vectors, P_π and P_λ representing probabilities of states. These will be updated during evaluation. Initially they are as follows. If N is a root node, the P_π holds its prior probabilities and the P_λ is not used. If N is a leaf, the P_λ is a unit vector and the P_π is not used. Otherwise, the P_π holds the probabilities of the node, averaged over all state combinations of its parents and the P_λ is a unit vector. Evidence is entered into a node by replacing its P_π and P_λ vector by one consisting of a 1 for the evidence state and 0's for all other states. Fig. 1 shows an example BN with its CPDs in Fig. 2. The P_π and P_λ vectors are given for each node, with no evidence.

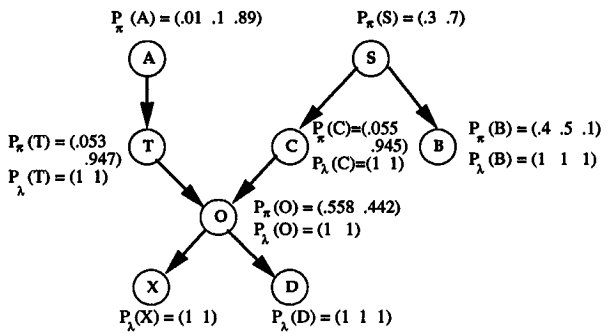

Figure 1: Example BN.

$$p(A) = (.01, .1, .89) \qquad p(S) = (.3, .7)$$

$$p(T \mid A) = \begin{pmatrix} .05, & .95 \\ .01, & .99 \\ .1, & .9 \end{pmatrix} \qquad p(C \mid S) = \begin{pmatrix} .1, & .9 \\ .01, & .99 \end{pmatrix}$$

$$p(B \mid S) = \begin{pmatrix} .5, & .4, & .1 \\ .3, & .6 & .1 \end{pmatrix} \qquad p(O \mid T, C) = \begin{pmatrix} .81, & .19 \\ .57, & .43 \\ .8, & .2 \\ .05, & .95 \end{pmatrix}$$

$$p(X \mid O) = \begin{pmatrix} .98, & .02 \\ .05, & .95 \end{pmatrix} \qquad p(D \mid O) = \begin{pmatrix} .6, & .2 & .1 \\ .45, & .5 & .05 \end{pmatrix}$$

Figure 2: CPDs for example BN.

Anytime Evaluation of BN

Our evaluation algorithm is a based on the message-passing polytree algorithm (Pearl 1988). As in the

polytree algorithm, each node N transmits π and λ messages which are real-valued vectors representing probabilities of states. Node N sends π messages to its parents and λ messages to its children. Consider a node X which receives π messages from parents $\mathbf{Z}$ and λ messages from children $\mathbf{Y}$. X updates its P_π and P_λ vectors as follows:

$$P_\lambda(X) = \prod_j \lambda_{Y_j}(X) \quad (1)$$

$$P_\pi(X) = \sum_i CPD_{X|\mathbf{Z}} \prod_i \pi_X(Z_i) \quad (2)$$

Using the messages received, node X computes new outgoing π and/or λ messages. See (Pearl 1988) for more details of the polytree algorithm. Each query node Q computes its belief by taking the dot product of P_π and P_λ vectors and normalising the result as follows:

$$BEL(Q) = \alpha\ P_\lambda(Q)\ P_\pi(Q) \quad (3)$$

where α is a normalising constant.

For each $Q \subset \mathbf{Q}$, the first result returned by our anytime algorithm is an initial estimate of the belief of Q obtained by collecting messages from its immediate neighbours. We then traverse the BN radially out from Q. Each visted node N will send a message to be propagated towards Q, which recomputes a more accurate estimate of its own beliefs as in Eq. 3. Only the branches on which visited nodes lie are re-evaluated. This corresponds to a partial evaluation of Eqs. 2 for N, with only messages from visited nodes being included.

By visiting a group of nodes, rather than a single node, at each iteration, we gather more information between successive computations of $BEL(Q)$. The question of how long to collect information and which nodes to visit at each stage is the meta-level control problem of *deliberation scheduling*, allocating computational resources between two stages (Dean & Wellman 1991, Ch.8).

The order of traversal is an important issue. Breadth-first and depth-first from the query node are obvious non-heuristic methods. Our *best-first* strategy is based on *arc weights* and also takes into account the *computational cost* of visiting a node. Weights of arcs at a node give an estimate of how much the node can be affected by each neighbour. We can also take into account the cost, in terms of computation time, of visiting a node. Ideally, we want to make a trade-off by visiting first the less costly but most influential nodes.

An Example

This example shows how the BN of Fig. 1 is evaluated, with evidence in node $X = 0$ and for query node O. We show the π and λ messages in Fig. 3 and the anytime evaluation results in Fig. 4. We first visit the immediate neighbours of O, to get an initial estimate, then the rest of the BN in depth-first order. In this scenario, $BEL(O)$ is re-computed after visiting each single node. The final result for the belief of O is (.733 .267), which is the exact answer. Note that nodes B and D are pruned because they are independent from the query node in this situation.

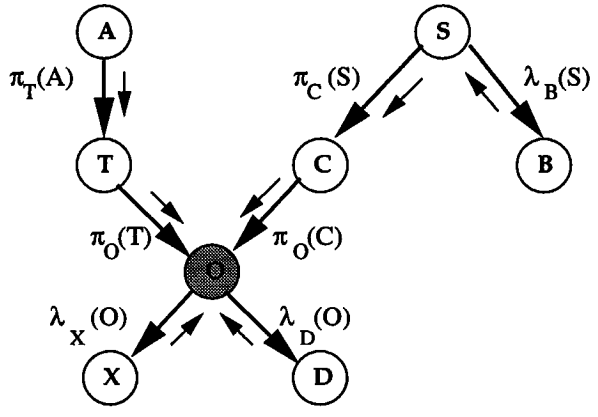

Figure 3: λ and π messages in BN with query O and evidence $X = 0$.

Node	Messages passed	$P_\pi(O)$	$P_\lambda(O)$	$BEL(O)$
T	$\pi_O(T)$ (.053 .947)	(.878 1.122)	(1 1)	(.439 .561)
C	$\pi_O(C)$ (.055 .945) $\pi_O(T)$ as above	(.117 .882)	(1 1)	(.117 .882)
X	$\lambda_X(O)$ (.98 .05)	(.117 .882)	(.98 .05)	(.723 .277)
A	$\pi_T(A)$ (.01 .1 .89) $\pi_O(T)$ (.091 .910) $\pi_O(C)$ as above	(.138 .864)	(.98 .05)	(.755 .245)
S	$\pi_C(S)$ (.3 .7) $\pi_O(C)$ (.037 .963) $\pi_O(T)$ as above	(.123 .877)	(.98 .05)	(.733 .267)

Figure 4: Anytime evaluation results of BN.

Arc Weight and Cost Measures for Best-First Graph Traversal

Arc Weights

The crucial aspect of incremental BN evaluation described above is the order of traversal of the BN. We want to visit those nodes which have the greatest influence on the query node first. If a node has several neighbours, we should first consider the one linked by the strongest connection. The weight of an arc provides an estimate of how much a change in the probability distribution of a node can affect the neighbour linked by this arc. Given a node Y with parent X and a

set of other parents $\mathbf{Z}$, we calculate the weight $w(X,Y)$ of the arc $X \rightarrow Y$ as follows:

$$w(X,Y) = tanh \frac{1}{\mid \Omega(\mathbf{Z}) \mid} \sum_{\mathbf{k} \in \Omega(\mathbf{Z})} \sum_{i,j \in \Omega(X)} D(p(Y \mid X = i, \mathbf{Z} = \mathbf{k}), p(Y \mid X = j, \mathbf{Z} = \mathbf{k})) \quad (4)$$

where $\Omega(N)$ is the set of states of node N, $\Omega(\mathbf{Z})$ is the set of combined states of nodes in $\mathbf{Z}$ and D is a distance between the two probability distributions. For this purpose, we use the Bhattacharyya (Bhattacharyya 1943) distance between two distributions P and Q. It is given by

$$D_B(P,Q) = -log \sum_i \sqrt{P_i Q_i} \quad (5)$$

We take the *tanh* in Eq. 4 to map the value to the range $(0,1]$. Eq. 4 calculates how much the probabilities of Y will change, as X changes state, averaged over all state combinations of all other parents. The weight is computed locally at each arc because only the values stored in the CPD for $p(Y \mid X)$ are required.

Since messages are also sent from child to parent of the original BN, a reverse measure of the weight of an arc is also needed. It is obtained by computing a CPD for $p(X \mid Y)$ using Bayes' Law and repeating the calculation.

In general, for nodes X_0 and X_n which are indirectly connected, we define the path weight $W(X_0, X_n)$ as:

$$W(X_0, X_n) = \prod_{i=0}^{n-1} w(X_i, X_{i+1}) \quad (6)$$

where X_{i+1} is the neighbour of X_i on the path to X_n. Taking the product over all arc weights on the path between X and X_n gives a measure of the *combined effect* of the intervening nodes and takes into account the distance between X and X_n, thus incorporating the impact of topological proximity and relevance described in (Poh & Horvitz 1996).

Note that our weight measure is suitable for multi-state nodes. In the case of binary nodes, it is simpler to use the *connection strength* measure introduced in (Boerlage 1995), from which ours was inspired:

$$CS(A,B) = \left| log \frac{p(b \mid a)}{p(\neg b \mid a)} - log \frac{p(b \mid \neg a)}{p(\neg b \mid \neg a)} \right| \quad (7)$$

Cost of Visiting a Node

We calculate the cost of visiting a node in terms of the number of additions and multiplications done when computing the π or λ message transmitted. This depends on the size of the CPD of the node. The cost of collecting a message from node X for its neighbour Y, which has parents $\mathbf{Z}$, is given by:

$$c(X,Y) = \begin{cases} \mid \Omega(Y) \mid \ (2 \mid \Omega(X) \mid \ - 1) \\ \quad \text{if } X \text{ sends a } \lambda \text{ message to } Y \\ \\ \mid \Omega(Y) \mid \ (2 \mid \Omega(\mathbf{Z}) \mid \ - 1) \\ \quad \text{if } X \text{ sends a } \pi \text{ message to } Y \end{cases} \quad (8)$$

In general, for nodes X_0 and X_n which are indirectly connected, we define the path cost $C(X_0, X_n)$ of sending a message from X_0 to X_n as:

$$C(X,Z) = \sum_{i=0}^{n-1} c(X_i, X_{i+1}) \quad (9)$$

where X_{i+1} is the neighbour of X_i on the path to X_n.

Best-First Traversal

When using cost and arc weights to traverse the BN, each arc is assigned a weight as defined in Eq. 4 and a cost is assigned to each node as in Eq. 8. To choose the best node to visit next for query Q, we scan all unvisited nodes directly connected to visited ones and select the node N with the largest ratio

$$\frac{W(N,Q)}{C(N,Q)} \quad (10)$$

In this way, we bias the search towards the node which most largely influences our query node, while making a trade-off with computational cost. Fig. 5 shows the cost and arc weights of our example BN, with query node O.

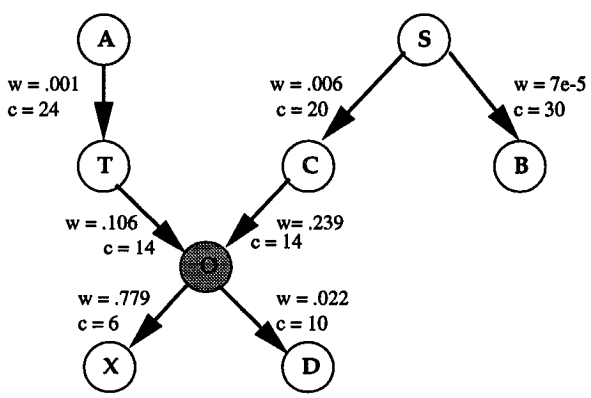

Figure 5: BN showing path weight and path cost for query O.

Experimental Results

The experimental results given in this section involve a polytree network of 22 nodes, shown in Fig. 6. It is adapted from the example in (Boerlage 1995). We evaluate the BN using breadth-first, depth-first and best-first strategies. We calculate an initial estimate of the posterior of our query node by visiting its direct neighbours. At each subsequent single node visit, we update the posterior of the query node and calculate the error in the result in terms of the Kullback-Leibler (KL) distance (Pearl 1988) between the current estimate and the exact value. Note that we initially evaluated the network using a join-tree algorithm (Lauritzen & Spiegelhalter 1988) to obtain the exact answer. For illustration purposes, we recomputed the posterior of the query node after every single node visit. Normally, it is preferable to visit a group of nodes at each iteration to minimise overheads and increase the rate of improvement in result accuracy.

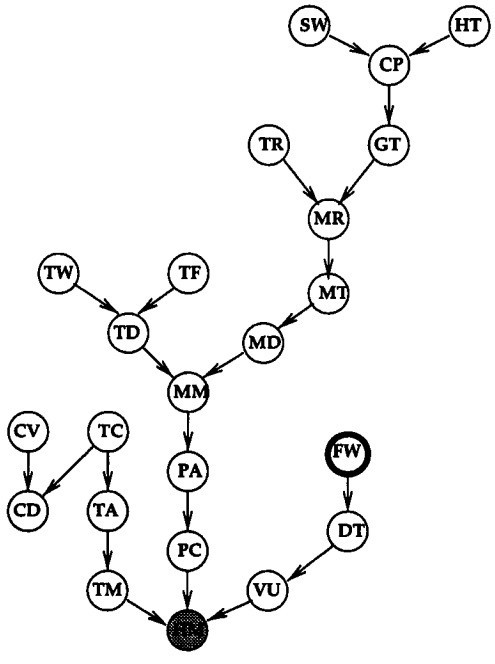

Figure 6: 22 node polytree, adapted from the example in [3], used to obtain experimental results. Query node is shaded and evidence node shown in bold.

Fig. 7 shows the performances of breadth-first, depth-first and best-first ($\frac{W(N,Q)}{C(N,Q)}$) strategies. **HM** is the query node and the evidence node is **FW**. The major dip in the curves occurs when node **PA** is visited. This large fall in the error shows that **PA** has a great influence on our query node. Because arc **PA** $\rightarrow$ **PC** has a large weight, our best-first strategy chooses to visit **PA** first. This is indicated by the early drop in the curve for arc weights curve.

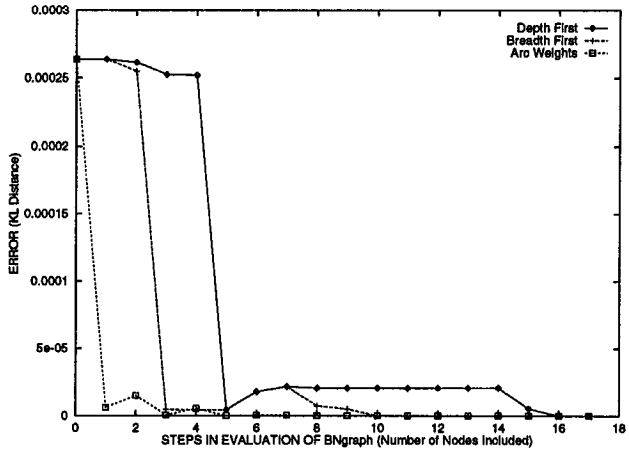

Figure 7: Comparison of graph traversal strategies: Breadth-First, Depth-First and Best-First (using Arc Weights).

Discussion

Belief networks have been used in applications that require real-time monitoring of complex domains, where the cost of computing an exact solution is prohibitive. Anytime algorithms have been developed for such problems. The approach described here regards BN evaluation as a search problem under limited resources. Evaluation is initially performed on a restricted number of nodes in the immediate vicinity of the query nodes. The BN is then traversed radially out from the query node and estimates for the belief of the query node are computed iteratively, i.e. after visiting a new node or group of nodes.

At each search step search, there is a result accuracy vs computation cost trade-off. The cost $C(N,Q)$ of visiting a node N and correspondingly updating the query node Q represents computational complexity. Our measure of path weights gives the strength of the connection $W(N,Q)$ between N and Q; this correponds to the improvement in the accuracy of our result. Our best-first strategy chooses to visit in priority node with the largest ratio $\frac{W(N,Q)}{C(N,Q)}$. Hence the search is biased in favour of more significant but less costly nodes.

We cannot obtain an exact run-time assessment of the solution quality at each step. But we have an estimate of any node's influence on the query node, in terms of the path weight. In general, visiting nodes on heavier paths gives rise to faster improvement in the result. Our cost measure can be used for run-time allocation of computational resources. With limited

time and memory at each step, we look at the cost of visiting some nodes to decide if the computation is feasible. Then, we want to select the most profitable group of nodes to include in our computation, given the amount of resources available. This is the meta-level control problem of deliberation scheduling.

Our current algorithm is a greedy search because it only chooses the next best option at each step. An issue to consider is how to perform a search with look-ahead, i.e. not just choosing the current best node, but the one leading to the most significant region of the BN, in terms of influence on the query node. For this purpose, we would need to develop a measure of the total weight of a section of the BN consisting of multiple branches diverging from a given node.

The procedure presented here does not strictly satisfy all the ideal requirements of anytime algorithms stated in (Zilberstein 1995). We can obtain a measurable quality of result only by comparing the output to the exact result which will not be available at run-time. While we can show empirically that in most cases, on average, the algorithm produces more accurate results over time, we cannot guarantee monotonicity. Our algorithm is fully interruptible. In this paper, we have shown how to apply our algorithm to BNs which are polytree structures. Since most BNs with underlying loops can be converted to clique-trees our formulation can be adapted to suit a larger category of structures.

References

Bhattacharyya, A. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematics Society* 35:99–110.

Boerlage, B. 1995. Link strengths in bayesian networks. Master's thesis, Dept. of Computer Science, U. of British Columbia.

Dean, T., and Wellman, M. P. 1991. *Planning and control.* San Mateo, Ca.: Morgan Kaufman Publishers.

Lauritzen, S., and Spiegelhalter, D. 1988. Local computations with probabilities on graphical structures and their applications to expert systems. *Journal of the Royal Statistical Society* 50(2):157–224.

Pearl, J. 1988. *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann.

Poh, K. L., and Horvitz, E. 1996. Topological proximity and relevance in graphical decision models. In *Proc. of 12th Conf. on UAI*, 427–435.

Zilberstein, S. 1995. Composition and monitoring of anytime alforithms. In *IJCAI 95: Anytime Algorithms and Deliberation Scheduling Workshop*, 14–21.