Techniques for Handling Complexity in Dynamic Belief Networks

Ann Nicholson and Stuart Russell

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-93-31
July 1993

Techniques for handling inference complexity in Dynamic Belief Networks

Ann Nicholson *
Department of Computer Science
Brown University
Providence, RI 02912
aen@cs.brown.edu

Stuart Russell
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

February 4, 1993

Abstract

Dynamic Belief Networks (DBNs) have been of interest recently as modelling tools for environments that change over time. In such networks, sets of nodes may be added automatically over time to represent current and future states. DBNs are typically multiply connected, and inference complexity is the biggest barrier to successful use. In this paper, we present both exact and heuristic techniques for state space reduction, thresholding and backtracking that can dramatically reduce inference costs without significantly reducing accuracy. We illustrate this in the domain of continuous monitoring of mobile robots. In order to control the costs and benefits of our heuristic methods, the system requires a good model of inference cost for any given network structure. We therefore also present results showing that Kanazawa's "join-tree cost" model gives a very good fit to actual inference costs in large networks.

Keywords: Bayesian networks, inference complexity, robot monitoring

*Much of this work was undertaken while the author was at the Robotics Research Group, Dept. of Engineering Science, Oxford University, supported by the Rhodes Trust. It has been continued at Brown University supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 and the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041.

1 Introduction

Belief Networks [Pearl, 1988], which integrate a mechanism for inference under uncertainty with a secure Bayesian foundation, were initially applied to fairly static domains, where the nodes and arcs do not change over time [Spiegelhalter *et al.*, 1989, Levitt *et al.*, 1989]. These approaches involved determining the network structure, supplying prior and conditional probabilities, adding or retracting evidence and repeating the inference algorithm for each change in the evidence. The complexity and size of networks for domains such as Natural Language Understanding motivated work on the dynamic *construction* of belief networks [Breese, 1989, Charniak and Goldman, 1989]. More recently, Dynamic Belief Networks (DBNs) (also called *Temporal Probabilistic Networks* [Dean and Kanazawa, 1989, Dean *et al.*, 1990], and *Dynamic Causal Probabilistic Networks* [Kjærulff, 1992]) have been of interest as modelling tools for environments that change over time [Kjærulff, 1992, Dagum *et al.*, 1992, Dean and Wellman, 1991, Andreassen *et al.*, 1991, Nicholson, 1992]. For such applications the network expands over time, as the state of each domain variable at different times is represented by a *series* of nodes.

Dynamic Belief networks have the following general characteristics. The nodes can be divided into three general categories: *world* nodes, which describe the central domain variables; *observation* nodes, which represent direct observations of world nodes, or the observable effects of a change in the state of a world node; *action* nodes, which represent the cause of a change in the state of a world node. Time is discretized; each time slice within the network represents the environment during that time interval, and the structure within time slices is usually very regular. Figure 1.1 shows the generic dynamic belief network structure for these node types; they are highly connected, particularly between time slices, and are Markovian, which constrains the state space to some extent.

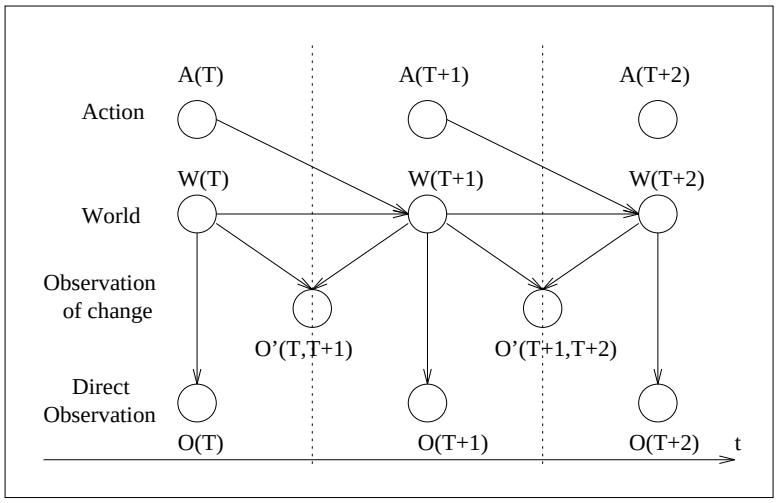

FIG. 1.1. Generic Structure for a typical Dynamic Belief Network.

In general, the complexity of inference for highly multiply-connected networks is NP-hard [Cooper, 1990]. Cooper [Cooper, 1990] suggests that the complexity problem should be tackled by designing efficient special-case, average-case and approximation algorithms, rather than searching for a general, efficient exact inference algorithm. Other researchers

who have encountered similar problems have described the use of domain specific knowledge and heuristics to reduce the search space. In addition, often a fixed window over the past, present and future is used, restricting the expansion of the network by maintaining a limited past, but regardless of the amount of ambiguity about that past, or its importance.

It is clearly impractical to maintain a complete history of the past, or to maintain all alternative world scenarios no matter how unlikely. Nodes representing the past must be deleted at some stage, and the total state space restricted. Such pruning decisions follow from the topological structure, and the current beliefs and assumptions, of the network. Therefore pruning can be implemented by domain independent meta-reasoning, which result in the reproduction of domain specific methods. Pruning may be exact (performed without loss of information), if there are sufficient certain positive or negative beliefs inferred from the network, or heuristic. For the latter, the problem of contradictions may arise later, in which case backtracking must take place; hence some sort of assumption structure must be retained. The basis of making a pruning decision is the tradeoff between the savings on execution of the inference versus the likelihood of making an error. Hence we need the ability to estimate the inference execution cost of networks, which we address in the following section.

2 Complexity and Performance: A Case Study

In previous work [Nicholson and Brady, 1992b, Nicholson and Brady, 1992a, Nicholson, 1992] we described in detail a Dynamic Belief Network (DBN) model for monitoring movement using light beam sensor data, showing how to represent trajectories, compare the observed trajectory against a scheduled trajectory, solve the data association problem, and handle incorrect data. The analysis of the performance of the DBN for this domain is the motivation for techniques for handling complexity described in the following sections. In Section 3 we describe a simplified version of the network which is sufficient to provide illustrative examples; the experimental results were obtained using the complete and considerably more complex network model described in detail in [Nicholson, 1992].

The network structure for our domain, which is typical of many dynamic applications, is highly multiply connected. This means that the standard, comparatively efficient inference algorithms for poly-trees are not applicable; intuitively, special techniques must be used to ensure that evidence is not counted twice. Various exact inference algorithms for this have been described, which fall broadly into two classes: conditioning and clustering [Pearl, 1988].

Our experimental results were obtained with an implementation of the DBN using the Jensen version of the Laurtizen-Spiegelhalter clustering algorithm provided by the IDEAL [Srinivas and Breese, 1989], currently the best algorithm available. [1] As further motivation for the techniques we will present for reducing DBN inference complexity, we consider in more detail the performance of this algorithm for our domain.

The Jensen algorithm first transforms the network into a *join tree*, then performs the inference on that join tree. We are interested in the "size" of the network and corresponding

[1] Although other inference algorithms for multiply-connected networks might favour different network topologies, given that the performance of the Jensen algorithm is much better than the others available, this restriction of attention to the Jensen algorithm seems quite reasonable.

performance of the inference algorithm. Kanazawa [Kanazawa, 1992] proposed the following *join-tree cost* to assess the computational expense of evaluating a probabilistic network.

$$\sum_{C_i \in C} \left(k_i \prod_{X \in C_i} |\Omega_X| \right)$$

where k_i is the number of distinct separation sets ($S_{ij} = C_i \cap C_j$, where cliques C_i and C_j are adjacent in the join tree) involving clique C_i. This cost is based on the number of multiplications required by the Jensen algorithm; the complexity of the Jensen inference algorithm is proportional to the product of the state spaces of the nodes in each clique.

Join-tree Cost: Experimental Results

First, we consider the relationship between the join-tree cost and the run-time for the Jensen inference algorithm. Figure 2.1 (a) and (b) shows the join-tree cost plotted against the run time for creating the join-tree and the run time for inference on the join tree respectively; these graphs use about 700 data points from running the Jensen inference algorithm on different combinations of structures, extension to the model, and domain parameters (see [Nicholson, 1992] for details). The results obtained were limited by the available computing resources. There is the usual trade-off between space and time; the Jensen algorithm runs comparatively quickly, but it takes up a lot of space. When the space requirements exceeded 40MB on a Sparc 2, the lisp process started thrashing, typically when the join-tree cost was 300,000 or more.

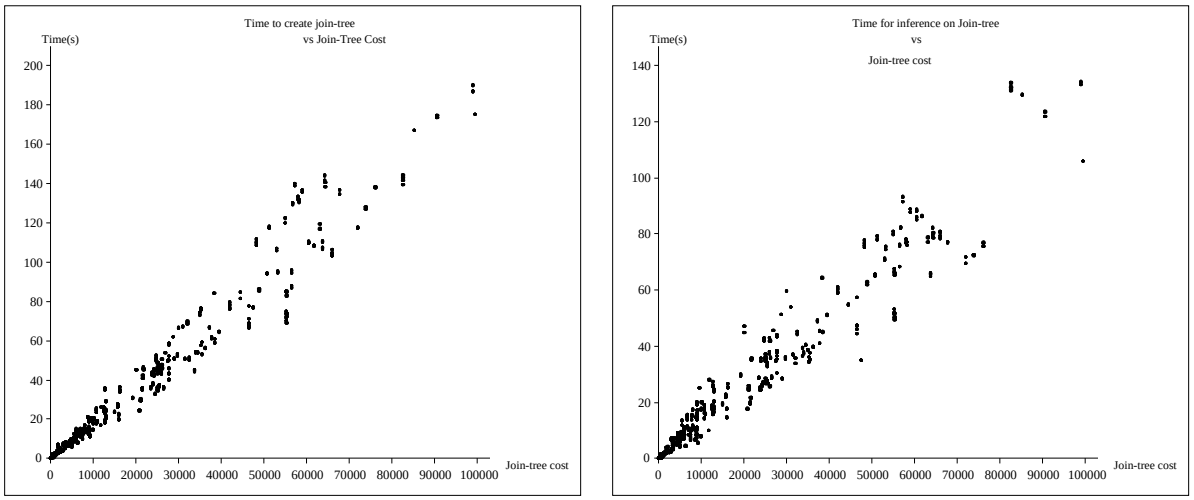

FIG. 2.1. Run time for (a) creating the join-tree, (b) join-tree inference, vs. Join-tree Cost.

The plots for the join-tree cost show a nearly linear relationship between run time and join-tree cost. This is empirical support of Kanazawa's theoretical result [Kanazawa, 1992], that the join-tree cost is a good measure of the cost of running the Jensen inference algorithm. While the correspondence applies to both the time to create the join-tree as well as running the inference on the join-tree, the join-tree need not be re-created in full each time, particularly when the changes in the network structure are small. [2]

[2]Of course, there may be more than one join-tree for a particular belief network. There is no p-time

3 Pruning

Pruning decisions follow from the topological structure, and the current belief and assumptions of the network. Therefore pruning can be implemented by domain independent meta-reasoning, which result in the reproduction of domain specific methods such as those by Dean *et al.* [Dean and Wellman, 1991, pp.330-334] among others. The aim of pruning is to reduce the state space of the network, and, in particular, to reduce the complexity of the inference algorithm, by getting rid of redundant or irrelevant information about the world. This is more flexible and general than maintaining a fixed window over the past, present and future. *Pruning* the network consists of any of the following actions:

- deleting states from a particular node;
- removing the connection between two nodes;
- removing a node from the network (i.e. removing all connections to and from a node).

Pruning case be exact (performed without loss of information) or heuristic. In this section we focus on identifying when exact pruning may be performed. In the following section, we show how we can do heuristic pruning by first making some assumptions, then applying the same pruning techniques.

Pruning the past once enough is known

In our domain all uncertainty is modelled within the DBN and evidence is never retracted from the network, therefore we can say certain things about the network beliefs. If the belief for a node being in a particular state becomes 1 or 0 at some stage, then this belief will not change. More formally, if $\texttt{belief}(\mathrm{V(T)}=\mathtt{s}_1) = 0$ or $\texttt{belief}(\mathrm{V(T)}=\mathtt{s}_2) = 1$, after the inference algorithm has been run at some time $\mathrm{T}_k > \mathrm{T}$, $\forall\ \mathrm{T}_k > \mathrm{T}$, $\texttt{belief}(\mathrm{V}(\mathrm{T}_k)=\mathtt{s}_1) = 0$, or $\texttt{belief}(\mathrm{V}(\mathrm{T}_k)=\mathtt{s}_2) = 1$, respectively. The system may then be considered to know something: in the former case, it knows that node V(T) is not in state $\mathtt{s}_1$; in the latter, it knows (the stronger fact) that node V(T) is in state $\mathtt{s}_2$. This knowledge, based on the internal network beliefs, is the basis for the pruning described in the remainder of this section.

It is possible to identify a subset of the *world* nodes, $\mathrm{W}_1(\mathrm{T})$, ..., $\mathrm{W}_q(\mathrm{T})$, within a time slice T, which represents either the entire world state, or the part we are interested in; we call these the *designated world nodes*. These nodes are chosen such that if their states are known, then nodes $\mathrm{V}(\mathrm{T}_k)$, where $\mathrm{T}_k < \mathrm{T}$, are no longer relevant to the overall goal of the inference.

Condition 1: $\forall i \in [1..q],\ \exists\ \mathtt{s}\ \texttt{belief}(\mathrm{W}_i(\mathrm{T}) = \mathtt{s}) = 1$

If Condition 1 holds then the general pruning action is as follows.

- Delete all nodes $\mathrm{V}(\mathrm{T}_k)$, where $\mathrm{T}_k < \mathrm{T}$.
- Explicitly incorporate knowledge that $\mathrm{W}_i(\mathrm{T}) = \mathtt{s}_i$.

---

algorithm which will always find the optimal join-tree (i.e. the join-tree with the smallest maximum cardinality). Kjærulff [Kjærulff, 1990] uses a process based on simulated annealing to choose the order in which to eliminate nodes. However, since the IDEAL system does not provide the facility to specify the order of node elimination, the results presented here are not necessarily for the best node ordering and we are unable to present comparative performance results showing the effect of varying the elimination order.

We explicitly incorporate the information that it is known to be in state $\mathtt{s}_i$ by deleting all states except for state $\mathtt{s}_i$, reducing the state space. We can also use negative information in a similar way. If a node V(T) has no successors and if $\mathtt{belief}(\mathrm{V(T)} = \mathtt{s}) = 0$, then we can delete state s.

We must also consider nodes in the time frame T other than those in the designated set, which shall be specified as $\mathrm{U}_1(\mathrm{T})$, ..., $\mathrm{U}_A(\mathrm{T})$. These nodes may have had predecessors from the previous time interval removed. If a node $\mathrm{U}_i(\mathrm{T})$ has other predecessors from within the same time interval, its conditional probability distribution must be reset; in many cases its remaining connections to predecessors will no longer have any valid causal meaning. In general, arcs from these predecessors within T should also be removed. If the state of a node $\mathrm{U}_i(\mathrm{T})$ was known before its predecessors were removed (i.e. the belief for one of its states was 1), this knowledge may be retained within the network. If the state of a node $\mathrm{U}_i(\mathrm{T})$ was not known when its predecessors were removed, the previous beliefs inferred by the DBN as represented by the belief vector can be used as the prior for this new root node; $\mathrm{P}(\mathrm{U}_i(\mathrm{T})|) = \mathtt{belief}(\mathrm{U}_i(\mathrm{T}))$. In this way information from the inference based on evidence in parts of the network now deleted is retained, to a certain extend although there may be a loss of information if there were conditional dependencies amongst the successors of deleted nodes. This method of node removal is very general; both Kjærulff's model reduction [Kjærulff, 1992] and the network reduction described in [Dean and Wellman, 1991], based on work by Shachter [Shachter, 1986], fall within the same framework.

Example of exact pruning

Our experimental domain is that of robot vehicle monitoring. The environment (a laboratory in which a robot vehicle roams) is divided into regions by the light beam sensors. The position of a moving object (person or robot) is given by the region it is in. Each light beam sensor provides data about a light beam sensor crossing (BC): the direction of the crossing, and the time interval over which it occurred. We assume that the number of objects, N, in the environment and the spatial layout of M regions and P sensors is fixed and known. Table 3.1 gives a summary of the types of nodes, their states, and their function in the network. The *world* nodes are those that represent the world state space: object position (OBJ) and heading (HEAD); the *action* nodes are the mobility (MOTION) nodes; the *observation* nodes are those representing the sensor crossings: a node for the crossing data provided by the sensor (BC-OBS), and a node representing the actual physical crossing of the sensor which occurred (BC-ACT). In [Nicholson, 1992] a detailed analysis of the network complexity for alternative network structures is described in terms of the domain parameters. In general the complexity for all the alternative structures is exponential both in the number of objects being monitored, and the number of regions.

In our domain, often the central interest is the positions of the objects, i.e the $\mathrm{OBJ}_i(\mathrm{T})$ nodes would be the designated world nodes. A very simple example of when ambiguity about the past is no longer relevant is illustrated in Figure 3.1, where there are 2 regions, 1 sensor and two objects (initial position unknown). The observed data are two successive `dir1` crossings. Assuming no uncertainty in the data, from this the DBN infers that both objects were initially in R_1 and at T_2 both objects must be in R_2. Unless the DBN receives additional evidence from another source, the uncertainty in the world state at T_1 will never

Node	States and Function	Node	States and Function
$\mathrm{OBJ}_i(\mathrm{T})$	$\mathtt{r}_1, \mathtt{r}_2, \ldots \ \mathtt{r}_M$ Position of object i, time T	$\mathrm{BC\text{-}OBS}_i$	`nc,dir1,dir2` crossing data from sensor i: `nc` indicates no crossing, `dir1`, `dir2`: the possible directions
$\mathrm{HEAD}_i(\mathrm{T})$	$\mathtt{u}, \mathtt{h}_N, \mathtt{h}_S, \mathtt{h}_E, \mathtt{h}_W$ Heading of object i: $\mathtt{u}$ = unknown;	$\mathrm{BC\text{-}ACT}_i$	`nc,dir1,dir2,both` Actual crossings of LB i; `both` represents objects crossing
$\mathrm{MOTION}_i(\mathrm{T})$	`stat, move` Mobility of object i; `stat` short for stationary in both directions		

TABLE 3.1. DBN node types for the monitoring domain.

be resolved. The domain specific condition for pruning is then $\forall i,\ \exists j$ $\mathtt{belief}(\mathrm{OBJ}_i(\mathrm{T}_2) = \mathtt{r}_j) = 1$. In general for this domain, if we know the positions of N_K of the N objects, then the state space of an OBJ node is reduced from M^N to $M^{N-(N_K)}$.

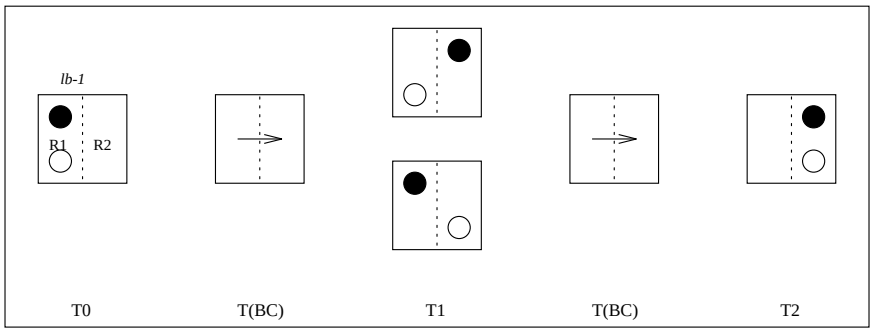

FIG. 3.1. Example with known current positions, but past ambiguity.

The basic DBN at time T_2 is shown in Figure 3.2. The nodes to be deleted are: $\mathrm{OBJ}_1(\mathrm{T}_0)$, $\mathrm{MOT}_1(\mathrm{T}_0)$, $\mathrm{HEAD}_1(\mathrm{T}_0)$, $\mathrm{OBJ\text{-}XED}_1(\mathrm{T}_1)$, $\mathrm{BC\text{-}ACT}_1(\mathrm{T}_1)$, $\mathrm{BC\text{-}OBS}_1(\mathrm{T}_1)$, $\mathrm{OBJ}_1(\mathrm{T}_1)$, $\mathrm{MOT}_1(\mathrm{T}_1)$, and $\mathrm{HEAD}_1(\mathrm{T}_1)$. Note that $\mathrm{OBJ\text{-}XED}_1(\mathrm{T}_2)$, $\mathrm{BC\text{-}ACT}_1(\mathrm{T}_2)$, and $\mathrm{BC\text{-}OBS}_1(\mathrm{T}_2)$ (the $\mathrm{U}_i(\mathrm{T})$ nodes) have not been deleted, but since all arcs from their predecessors have been deleted, they are no longer connected to the rest of the network and evidence about them cannot affect the rest of the network.

Propagating the effects of the initial pruning

Suppose that a node V(T) has no predecessors, and its state is known to be $\mathtt{s}_i$. When its other states $\mathtt{s}_j \neq \mathtt{s}_i$ are deleted the conditional probability tables of its successors nodes must be updated to reflect this. Take a successor node U, which must be an instance for either T, or T+1. Any probability distribution entry of the form $\mathrm{P}(\mathrm{U}{=}\mathtt{s} |\ \ldots\ \mathrm{V(T)}{=}\mathtt{s}_j \ldots) = x$, where $\mathtt{s}_j \neq \mathtt{s}_i$, must be deleted, since $\mathtt{s}_j$ has been deleted. After such a revision of the conditional probability table for U, there may be states of U which are impossible, as defined by: $\forall$ Conditioning Cases `cc`, s.t. $\mathrm{P}(\mathrm{U}{=}\mathtt{s}_{impossible} |\ \mathtt{cc}) = 0$. All such states $\mathtt{s}_{impossible}$ may then be deleted, and the pruning procedure performed recursively on U's successors.

During subsequent network expansions, the network construction algorithm can use the same check on the conditioning cases to remove impossible states as new nodes are added.

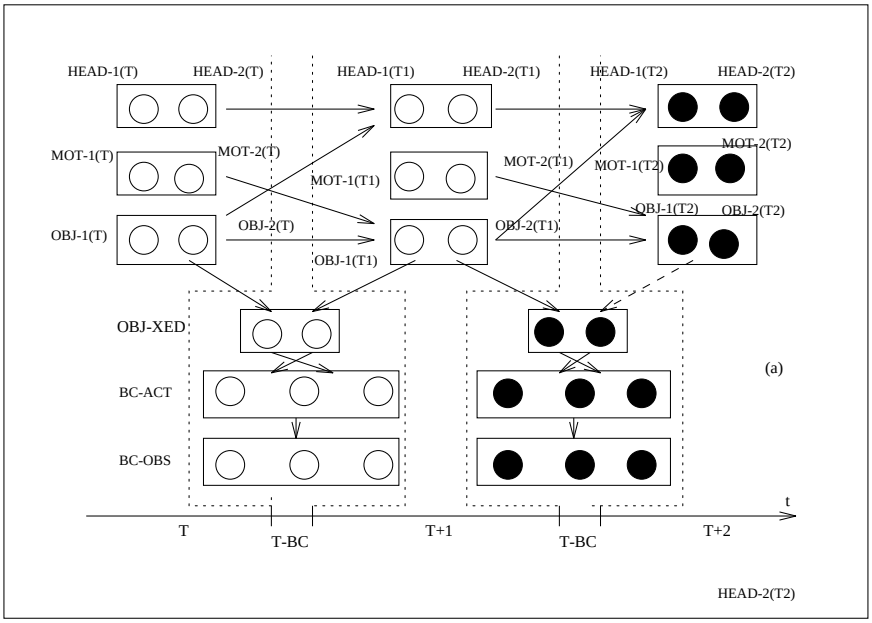

FIG. 3.2. DBN for past ambiguity example shown in Figure 3.1. The filled nodes are those remaining. Arcs shown by a dashed line also deleted.

After the pruning propagation, a root node whose state is known could also be removed from the network without affecting the inference in the remainder of the network. Since we do not want to lose the information contained in the node (i.e. its state), the pruning process deletes the connections to its successors. It is therefore useful for the underlying system to allow networks to be not fully connected; the IDEAL system does, the current implementation of HUGIN does not. However, this disconnecting of a known root node provides minimal improvement in the performance, since it has only one state; this has not been implemented for the results described below.

Example of pruning propagation

Suppose the environment consists of M regions, in a linear arrangement, containing a single object known to be in R_j at time T: $\texttt{belief}(\mathrm{OBJ(T)} = \mathtt{r}_j) = 1$. Pruning this known root nodes cuts the state space size from M to 1. The only non-zero conditioning cases (ignoring heading and assuming movement in either direction is equiprobable) for OJB(T+1) are:

$$\mathrm{P(OBJ(T+1)}{=}\mathtt{r}_j|\ \mathrm{MOTION}_i\mathrm{(T)} = \texttt{stat}, \mathrm{OBJ(T)}{=}\mathtt{r}_j){=}1$$
$$\mathrm{P(OBJ(T+1)}{=}\mathtt{r}_{j+1}|\ \mathrm{MOTION}_i\mathrm{(T)} = \texttt{move}, \mathrm{OBJ(T)}{=}\mathtt{r}_j){=}0.5$$
$$\mathrm{P(OBJ(T+1)}{=}\mathtt{r}_{j-1}|\ \mathrm{MOTION}_i\mathrm{(T)} = \texttt{move}, \mathrm{OBJ(T)}{=}\mathtt{r}_j){=}0.5$$

Using the pruning propagation described in the previous section, all states except for $\mathtt{r}_{j-1}$, $\mathtt{r}_j$, and $\mathtt{r}_{j+1}$ are deleted from OBJ(T+1). The effect of this exact pruning is to reduce the states of OBJ_i(T+1) to the known previous position or adjacent positions, using the physical constraints represented in the conditional probabilities.

Experimental results

We tested the propagated pruning technique on a range of network structures, domain parameters and example environments for the monitoring domain. We assumed that we knew the position of the objects at time T_0, reduced the state space for those root nodes, and then propagated the effect of the pruning. Let C_{orig} be the join-tree cost for the original network and C_{pruned} be the join-tree cost for the pruned network. Figure 3.3 shows a graph of the results obtained as a plot of the ratio of the C_{orig}/C_{pruned} against C_{orig}. The ratio increases linearly, indicating that the propagated pruning provides superlinear reduction in the join-tree cost, and hence in the inference time. For networks where the join-tree cost approached 100,000, the reduction ratio varied from about 20 up to 220.

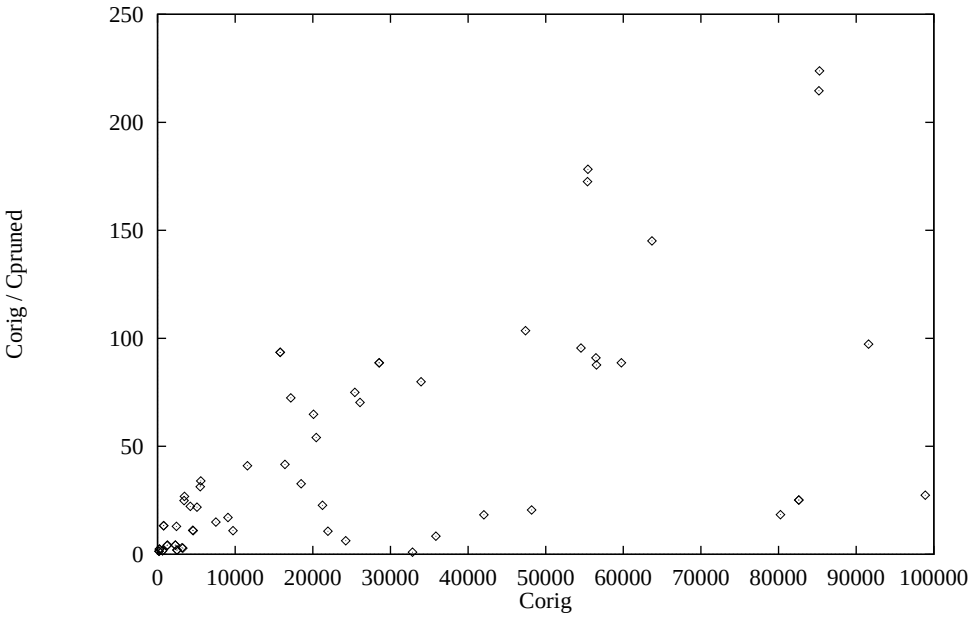

FIG. 3.3. Results of pruning algorithm.

No performance results are given showing the cost of the pruning process itself since the current implementation is non-optimised, however it is clear that the cost of the pruning process is more than offset by the gains in the performance of the inference.

4 Assumptions and Pruning

The exact pruning results are encouraging, however the pruning technique just described is not sufficient for controlling the inference complexity; if there is no initialisation information, no pruning will be performed, and even with initialisation information, if the DBN models sensor failure then there is always a small chance that the data is incorrect. This means that at no subsequent time frame will the state of all designated world nodes $W_1(T)$, ..., $W_q(T)$ be known, and no further pruning will be performed. Assumptions about the world and the observation data are required.

Consider the situation when driving down a highway, where a driver usually begins with the assumption that the car's speedometer is providing the correct speed. She then uses

observations about how fast and often she is passing other cars to infer the speed of other cars on the road. Suppose the speedometer reading is 55 m.p.h, and the driver notices that she is passing every other car on the road quite rapidly. The resulting inference that all the other cars on the road are travelling at 35 m.p.h or less contradicts another belief about the world, i.e. that cars usually travel at about 60 m.p.h on highways. In this case, the driver must recall the original assumption that the speedometer is working and question its validity.

The basis of making an assumption or pruning decision is the tradeoff between the savings on execution of the inference versus the likelihood of making an error. In this section we outline a method of making assumptions using thresholds which can be used in conjunction with the pruning mechanism described in the previous section.

Making Assumptions

The DBN may maintain a number of alternatives, most of which are quite unlikely. There are two kinds of assumptions which may be made. A *negative assumption* is made when an alternative is removed which the DBN infers has possibility below a certain threshold τ_{min}. A *positive assumption* is made when committing to an alternative which the DBN infers has possibility above a certain threshold τ_{max}. This definition of assumption is deliberately general; in particular, an alternative may be for an individual node, it may be for the joint distribution specified by the network as a whole, or it may fall somewhere in between these two; we will call this the *assumption set.*

Making assumptions satisfies the necessary conditions for activation of the pruning process described in the previous section. We identify the following issues which a thresholding assumption must address.

- Should τ be a function of the individual node, i.e $\tau(N)$, where N is any node in the network??
- How many and which of the candidate nodes (those node which have a belief over their threshold) should form a single assumption set?

We can derive obvious heuristics based on the network topology and the nature of both the pruning procedure and the join-tree cost. Root nodes and the designated world nodes should be given priority for being thresholded. The former will lead to maximum propagated pruning and state space reduction, while thresholding all the designated world nodes for a given time slice will result in pruning the past before that time. Within the sets of root and designated world nodes, nodes which are in the larger cliques should be given higher thresholding priority, as this will have a greater impact on the join-tree cost (and hence the inference performance) than thresholding nodes which are only part of a small clique.

Handling subsequent contradictions

The problem then arises as to what to do if the assumptions made by the thresholding procedure are incorrect and later observations give rise to a contradiction or infer a very unlikely situation. The system must maintain a record of the assumptions being made. For each node V(T) in an assumption set, we know that

`belief`(V(T)=`s`) = bel, where $bel > \tau_{max}(V(T))$

The data structure for storing the assumption information takes the form: (V(T), `s`, bel). Rather than recording all the individual changes that are made when pruning after making an assumption, it is simpler to store a copy of the original DBN (the size requirements are much less than the join-tree, and it need not be kept in main memory).

Let us consider when earlier assumptions are questioned and how the assumption sets to be retracted are chosen. If there is no uncertainty in the data, the simple DBN infers a contradiction and the problem is then to measure how certain we were of a particular assumption set. If the assumption set consisted of the nodes V_1 to V_k, where `belief`(V_i = `s`$_i$) = $bel_i > \tau$, then this measure should be function of the beliefs bel_i, i.e. $f(bel_1, bel_2, ..., bel_k)$.

If the network internally models uncertainty about the data, the network will not necessarily infer a contradiction. For example, in our monitoring domain, the DBN infers that a sensor is defective and this is in fact used for sensor validation [Nicholson and Brady, 1992b]. We therefore require a measure of the likelihood of new evidence. For the monitoring domain, we want to capture the intuition that it is more likely an earlier assumption was incorrect than that *all* the sensors are malfunctioning. In [Nicholson, 1992] a method is described for determining the difference between predicted and observed beliefs within the network. However, the real problem with this method is that the prediction is based on the assumption set. We require some measure of the probability that the observations would have occurred without setting the assumptions. This remains an open research question.

General dynamic network construction algorithm

We can now outline a general dynamic network construction algorithm, an updated version of that given in [Nicholson, 1992], currently unimplemented, which includes both using thresholds for making assumptions, and pruning.

1. Expand DBN
2. Add data evidence
3. Run inference algorithm
4. While there exist contradictions (certain or likely) do
   - **(i)** choose assumption set to retract
   - **(ii)** undo pruning (retrieve stored DBN,include subsequent DBN expansions and evidence)
   - **(iii)** retract assumptions
   - **(iv)** run inference algorithm
5. Do thresholding, recording assumption set.
6. Make copy of DBN
7. Do pruning
8. Go to 1.

5 Conclusions

Inference complexity is the biggest barrier to successful use of Dynamic Belief Networks to model environments that change over time. We have presented both exact and heuristic techniques for state space reduction through pruning, thresholding and backtracking that can dramatically reduce inference costs without significantly reducing accuracy. We have shown experimental results from the domain of continuous monitoring of mobile robots demonstrating that Kanazawa's "join-tree cost" model provide a good model of inference cost for any given network, and illustrating the reductions in the join-tree cost obtained by propagated pruning.

We have limited our focus to techniques for improving the complexity of an exact inference algorithm, by focusing on simplifying or reducting the model. However these techniques are orthogonal to the use of approximate inference algorithms [Henrion, 1988, Shachter and Peot, 1989, Shwe and Cooper, 1990, Horvitz *et al.*, 1989], so the effect of combining pruning with approximation algorithms should be multiplicative.

The problems of how to assess the correctness of an earlier approximation or assumption in the light of new evidence, and what action to take if the approximation or assumption is deemed likely to be incorrect, are yet to be addressed. However we have shown that using meta-level reasoning during the construction of a belief network can improve significantly the performance of the inference.

References

[Andreassen *et al.*, 1991] Andreassen, S.A.; Benn, J.J.; Hovorks, R.; Olesen, K. G.; and Carson, R. E 1991. A probabilistic approach to glucose prediction and insulin dose adjustment: Description of metabolic model and pilot evaluation study. Unpublished draft.

[Breese, 1989] Breese, J.S. 1989. Construction of belief and decision networks. Technical Memorandum 30, Rockwell Palo Alto Laboratory, 444 High Street, Palo Alto, California 94301.

[Charniak and Goldman, 1989] Charniak, Eugene and Goldman, Robert 1989. Plan recognition in stories and in life. In *Proc. of the Fifth Workshop on Uncertainty in Artificial Intelligence.* 54–60.

[Cooper, 1990] Cooper, G.F. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence* 42:393–405.

[Dagum *et al.*, 1992] Dagum, P.; Galper, A.; and Horvitz, E. 1992. Dynamic network models for forecasting. In *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence.*

[Dean and Kanazawa, 1989] Dean, Thomas and Kanazawa, Keiji 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5:142–150.

[Dean and Wellman, 1991] Dean, Thomas and Wellman, Michael P. 1991. *Planning and control.* Morgan Kaufman Publishers, San Mateo, Ca.

[Dean *et al.*, 1990] Dean, Thomas; Kanazawa, Keiji; and Shewchuk, John 1990. Prediction, observation, and estimation in planning and control. In *Proceedings of the Fifth IEEE International Symposium on Intelligent Control.* 645–650.

[Henrion, 1988] Henrion, M. 1988. Propagating uncertainty in bayesian networks by logic sampling. In Lemmer, J. and Kanal, L., editors 1988, *Uncertainty in Artificial Intelligence Vol 2.* North Holland, Amsterdam. 149–163.

[Horvitz *et al.*, 1989] Horvitz, E.J.; Suermondt, H.J.; and Cooper, G.F. 1989. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Proc. of the Fifth Workshop on Uncertainty in Artificial Intelligence.* 182–193.

[Kanazawa, 1992] Kanazawa, Keiji 1992. *Probability, Time, and Action.* Ph.D. Dissertation, Brown University, Providence, RI.

[Kjærulff, 1990] Kjærulff, U. 1990. Graph triangulation — algorithms giving small total state space. Technical Report R 90-09, University of Aalborg, Denmark.

[Kjærulff, 1992] Kjærulff, U. 1992. A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence.*

[Levitt *et al.*, 1989] Levitt, T.S.; Agosta, J. M.; and Binford, T.O. 1989. Model-based influence diagrams for machine vision. In *Proc. of the Fifth Workshop on Uncertainty in Artificial Intelligence.* 233–244.

[Nicholson and Brady, 1992a] Nicholson, A. E. and Brady, J. M. 1992a. The data association problem when monitoring robot vehicles using dynamic belief networks. In *Proc. of the 10th European Conf. on Artificial Intelligence (ECAI-92).*

[Nicholson and Brady, 1992b] Nicholson, A. E. and Brady, J. M. 1992b. Sensor validation using dynamic belief networks. In *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence.*

[Nicholson, 1992] Nicholson, A. E. 1992. *Monitoring Discrete Environments using Dynamic Belief Networks.* Ph.D. Dissertation, Dept. of Engineering Sciences, Oxford University.

[Pearl, 1988] Pearl, Judea 1988. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Mateo, Ca.

[Shachter and Peot, 1989] Shachter, R. and Peot, M. 1989. Simulation approaches to general probabilistic inference on belief networks. In *Proc. of the Fifth Workshop on Uncertainty in Artificial Intelligence.* 311–318.

[Shachter, 1986] Shachter, R.D. 1986. Evaluating influence diagrams. *Operations Research* 34:871–882.

[Shwe and Cooper, 1990] Shwe, M. and Cooper, G. 1990. An empirical analysis of likelihood-weighting simulation on a large, multiply connected belief network. In *Proceedings of the Sixth Workshop on Uncertainty in Artificial Intelligence.* 498–508.

[Spiegelhalter *et al.*, 1989] Spiegelhalter, D.; Franklin, R.; and Bull, K. 1989. Assessment criticism and improvement of imprecise subject probabilities for a medical expert system. In *Proc. of the Fifth Workshop on Uncertainty in Artificial Intelligence.* 335–342.

[Srinivas and Breese, 1989] Srinivas, Sampath and Breese, Jack 1989. Ideal: Influence diagram evaluation and analysis in lisp. Technical Report Technical Memorandum No. 23, Rockwell International Science Center.