ELSEVIER

Artificial Intelligence 107 (1999) 99–124

Artificial Intelligence

Fast Bayes and the dynamic junction forest

J.Q. Smith [a,*], K.N. Papamichail [b,1]

[a] *Department of Statistics, University of Warwick, Coventry, CV4 7AL, UK*
[b] *School of Informatics, University of Manchester, Manchester, M13 9PL, UK*

Received 10 April 1997; received in revised form 11 June 1998

Abstract

It has been shown that junction tree algorithms can provide a quick and efficient method for propagating probabilities in complex multivariate problems when they can be described by a fixed conditional independence structure. In this paper we formalise and illustrate with two practical examples how these probabilistic propagation algorithms can be applied to high dimensional processes whose conditional independence structure, as well as their underlying distributions, are augmented through the passage of time.

Keywords: Dynamic models; Hellinger metric; Influence diagrams; Junction trees; Probabilistic expert systems; Multivariate state space models

1. Introduction

Over the last ten years it has been established that Bayesian expert systems can be made to work quickly and accurately on a wide range of practical problems provided that the underlying relationships between variables in the problem remained fixed. These relationships are often represented by influence diagrams or causal networks—see [2,8,12, 26,31] and later in this paper. However, there are many problems where such relationships evolve; for example, the conditional independencies coded in the graph may be eroded as information is gathered. Alternatively, the structure of dependencies between variables may change through the passage of time. Indeed in any learning environment we believe that such an evolution of structure is almost inevitable—so machine learning at some point needs to address this issue. However, until now this has been largely ignored—for an exception see Kjærulff [11]. Learning in a dynamic environment is fundamentally different

* Corresponding author. Email: j.g.smith@warwick.ac.uk.
[1] Email: nadia.papamich@man.ak.uk.

0004-3702/99/$ – see front matter
PII: S0004-3702(98)00103-9

from relationships that are static in a number of key ways as it can be illustrated through the following example. Suppose a company produces a number of different brands of products within a large market and knows all her competitors' brands. There are now marketing techniques available to enable that company to produce a causal network over the vectors of sales made within a given short time interval [19,20]. This causal network codes up the dependency structure between the sales of different brands at that time and after being transformed into a junction tree [9] it can be used to update efficiently the probability distributions of sales of one product given precise information about collections of others. These quick algorithms are now well understood in a static environment where the causal network remains unchanged over time—for a helpful introduction see Jensen [9]—and are outlined in the next section.

However, in a dynamic environment we encounter two problems. Firstly, the algorithms of Jensen cannot usually be employed directly because the company's database will not be complete enough to observe directly the variables in the causal network. Instead it will typically have incomplete noisy data which is not fully disaggregated into sales information on the individual brands in the network. For example, a company will often have much better information about its own product sales than those of its competitors. Unfortunately, these types of imperfection can induce other dependencies in the system. The simplest example of this is when we acquire perfect information about the sum S of the sales of two brands whose sales $S(1)$ and $S(2)$ are not dependent, being in different sections of the market. Having observed S, knowing $S(1)$ tells us precisely the value of $S(2)$. So the aggregate S induces a dependence between $S(1)$ and $S(2)$. The causal network and possibly also its junction tree will therefore need to be adjusted in the light of this piece of information. Another type of dependence, induced by missing data, is discussed in [31]. Of course a different causal network and its associated junction tree might be drawn to include all possible aggregates, but a dynamic environment is not really conducive to such pre-processing. It should always be possible to absorb new information as it becomes available even if it is not in the pre-processed format. It is therefore highly desirable to adjust the inferential framework derived from the influence diagram as and when it becomes necessary.

A second problem generic to dynamic environments like the one above is that it is often possible that components and the underlying dependence structure will periodically change. In the application above, for example, a competitor may introduce a new competing brand. Alternatively she may choose to reprise or re-advertise and therefore reposition a brand so that it has a different set of competitors. In this case, the influence diagram which represents the dependency structure will need to change.

In Section 4 of this paper we will formalise the description of a dynamic system in terms of a sequence of graphs called junction forests. This formalism is based on the authors' experiences in developing algorithms in two different fields of application: the first, the forecasting of product sales in a competitive market [21,22] and the other the forecasting of the spread of contamination after a nuclear accident [4,6,29]. The second application has been coded up as an operational dynamic junction forest [4].

In Section 5 we specify an algorithm for automatically adjusting both the graph on which calculations are made and also the objects (the cliques) which form the nodes of that graph. This allows a propagation algorithm—originally designed to work when each observation

relates only to states in a single clique of the original forest—to be valid generally. Such adjustments often reduce the subsequent efficiency of the system because the new junction forest usually has a smaller number of cliques of a larger size.

In Section 6 we introduce several transformations of a junction forest that automatically changes the representation of the joint probability distribution so that it is a more compact and efficient form. The method modifies not only the number of states contained in each clique but also the relationships between them and the existence of the cliques themselves, discarding variables no longer of interest. In Section 7 we outline how approximate techniques can also be incorporated within such a system giving an example of such a transformation used in conjunction with the Hellinger metric between a density and its approximation. This section concludes with an illustration using the nuclear example described in Section 3.

We begin the paper with a short review of state space representations of problems and the junction tree propagation algorithm [9,10].

2. From dynamic influence diagrams to junction graphs

For simplicity and only with a slight loss of generality, we shall assume that our dynamic system is defined in discrete time. One standard class of discrete time state space models (see, for example, [7,34]) is specified in terms of an observation equation and a system equation respectively:

$$Y_t = F_t\theta_t + \upsilon_t, \tag{2.1}$$

$$\theta_t = G_t\theta_{t-1} + \omega_t. \tag{2.2}$$

An illustration of the above model is given in Fig. 1.

The error distribution of $\{\upsilon_t\omega_t\colon\ t = 1, 2, \ldots\}$ is nearly always chosen to be mutually independent so that an observation Y_t at time t satisfies the conditional independence statement:

$$Y_t \coprod \theta_1, \theta_2, \ldots \mid \theta_t, \tag{2.3}$$

i.e., an observation at time t only gives direct information about the values of components of the state vector at that time, and

$$\theta_{t+1}, \theta_{t+2}, \ldots \coprod \theta_1, \ldots \theta_{t-1} \mid \theta_t, \tag{2.4}$$

i.e., states are Markov in time—we only need to retain our beliefs about the current state vector and then we can determine any probability statements we might wish to make about the future.

The random vectors $\theta_t = 1, 2, \ldots$ are called state random vectors. In dynamic systems the state vector is an important object because if we were given its value we could disregard

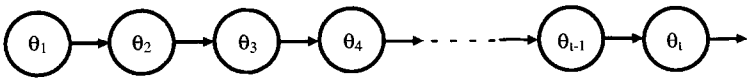

Fig. 1. A discrete time state space model.

all other information as irrelevant for predicting future observations Y. In this sense the state vector contains all the relevant information we need for forecasting. In our simple marketing example, the state random vectors will be the vectors of sales of each brand at time t, which typically will not be known precisely by the company. The matrix G specifies how these sales are expected to develop in time. The simplest evolution would make G the identity matrix, which would stipulate that sales of each brand in the time period t were expected to be the same as they were in the time period $t-1$. The error vector ω_t, captures the likely random drift in this relationship. Each vector Y_t, $t = 1, 2, \ldots$, is the set of possibly incomplete data received in the interval t where we can read from the rows of the matrix F the aggregate of brands associated with each observational component of Y_t. There will usually be some measurement error associated with these observations and the vector υ_t just represents this.

A useful way of coding conditional independence statements $(x_1, \ldots, x_n)$ is by a causal network (graph of an influence diagram) as a set of random vectors [9]. Given a collection of conditional independence statements of the form:

$$x_k \coprod \Omega_k \mid \Pi_k \quad 2 \leqslant k \leqslant n, \tag{2.5}$$

where Ω_k, Π_k are disjoint subsets of $\{x_1, x_2, \ldots, x_{k-1}\}$ whose union $\{x_1, x_2, \ldots, x_{k-1}\}$ is a causal network, i.e., a directed acyclic graph whose nodes are $x_1, \ldots, x_n$ with a directed edge from x_i to x_k, $1 \leqslant i \leqslant k \leqslant n$, if and only if x_i lies in the set Π_k.The elements of Π_k are called the *parents* of x_k. A directed acyclic graph is a *valid causal network* if and only if all the conditional independence statements of Eq. (2.5) are true. It is easy to show that a valid causal network on the nodes $\theta_1, \theta_2, \ldots, \theta_t$ associated with the system equation (2.2) has the form of Fig. 2. Deductive rules on causal networks/influence diagrams have been well studied—see, for example [13,14,17,24].

In our context, in order to develop the most powerful methods of learning on causal networks it will be important to allow nodes to represent individual components of a state vector at any given time. To illustrate this we return to our marketing example.

In time period 1, there are eight brands in the market $\{A, B, C, D, E, F, G, H\}$. Suppose that from expert judgment it is possible to asset that the sales of the three collections of brands $\{A, B, C, D\}$, $\{E, F\}$ and $\{G, H\}$ are independent of each other and given the sales of B, the predictions of sales of brand C will not depend on anything learnt about A and the prediction of brand sales of D will not depend on anything learnt about the sales of A and C. It is straightforward to show that these judgments can be cited in the causal network of Fig. 2.

Now we can use the system equation to extend the causal network to depict the dependency structure when we add nodes representing sales at future times. For the sake

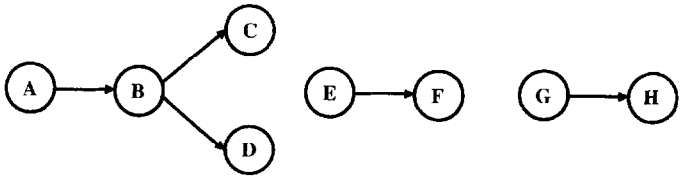

Fig. 2. A causal network between brands sales in the first time period.

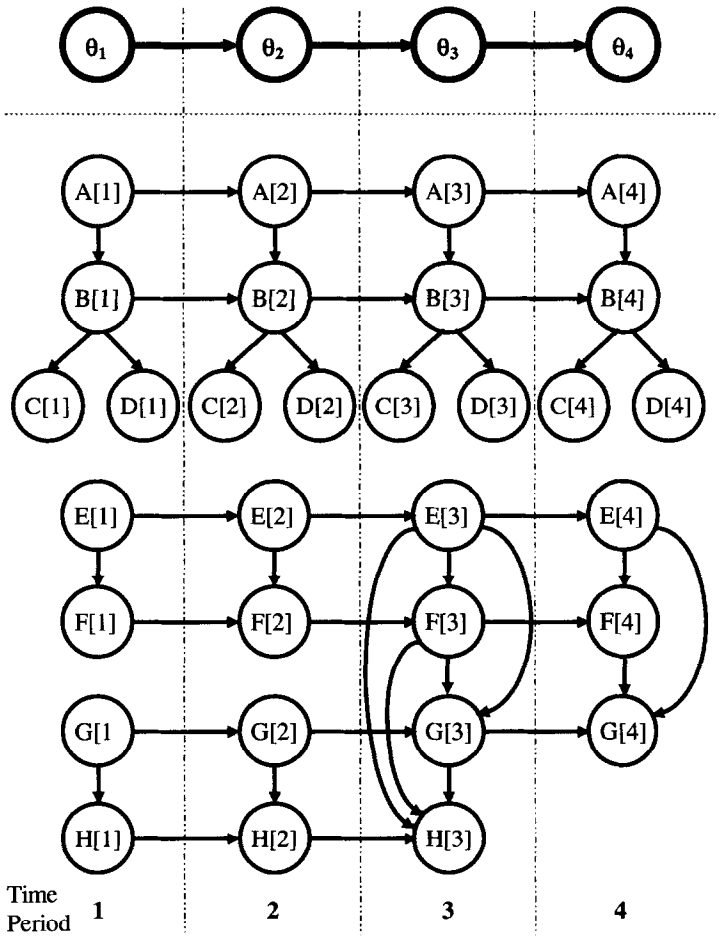

Fig. 3. A causal network of components of states associated with the levels of sales in the first four time periods when no data is collected and errors in components of ω_t, $t = 1, 2, \ldots$, are all independent.

of simplicity we shall assume a diagonal system matrix G with zeros in the diagonals in the third and fourth rows and independent components to the system error terms. Given that the same competitive structure of the market is valid in the second time period, it is easy to check that a valid causal network relating brand sale over the first two time periods is given by the corresponding subgraph of the graph of Fig. 3. During time period 3 we have assumed that several advertising companies have induced brands $\{E, F\}$ and $\{G, H\}$ to compete against one another. They are therefore predicted to become dependent and this is represented in the subgraph associated with the third period. Within time period 4, brand H is withdrawn which gives rise to the adjusted pattern over this period. The full causal network of this process is given in Fig. 3.

There are various issues highlighted by this example. First, even in small problems like the one above, the number of variables increases in time so that the necessary probability

manipulations can become more and more involved. However, if we have a Markov structure and we only need to make predictions into the future then the situation can be retrieved. This is because, by storing only the probability distributions over the current states given all the data collected to date, we have all the information we need from the past to enable us to estimate into the future. Thus, in this context if our current data of a subset of the system's variables can make valid predictions it is sufficient to remember the causal network and its associated probability distribution. A description of how this can be achieved is given later in the paper.

Let us return to our example. For time period 1, the causal network of Fig. 2 is valid. By using a distribution over these variables together with the observation and system equations above it is possible to calculate any probability distribution over future events. Now suppose we observe sales figures about A, B, C, D, $E+F$, G and H with independent errors. Using well known results (see Spiegelhalter et al. [31] and later in the paper) it is easy to show that we obtain a new distribution over these variables while the causal network remains valid. It is also possible to check directly from the d-separation theorem [9,17] that time distribution for sales in the second period given this information follows the same network. Again by using the observation and system equations we are able to predict all probabilities about future events from the distribution over this network. The data from the second period consists of observations with independent measurement errors of A, B, $C+D$, E, F, G and H. It can be shown that since we do not observe C or D separately but only obtain a noisy estimate of their sum a dependence is introduced between C and D and this dependence needs to be carried forward into the time period 3. This is a special case of results discussed in Section 5 of this paper. The dependence between E, F, G and H induced by advertising activity is carried through on to the other part of this marginal network. The data with independent measurement error in period 3 is on A, B, C, D, $E+F$ and G; product H having been withdrawn. A valid network for the variables in time 4 can now be shown to be the one given in Fig. 4.

Because of the Markov assumption all probabilities about future sales in our example depends on our data only as it has affected the distribution of current states; not past ones. It follows that all we need to retain is our distribution on current states. Past states will only be important if by retaining them our probability propagation algorithms can be made significantly quicker.

In this paper we investigate how to efficiently code the joint distribution of the subvector ϕ_T of $(\theta_1, \theta_2, \ldots, \theta_t)$ which in a particular application, will be sufficient for the predictive requirements of the model. In our example this vector ϕ_T was just θ_t. In the environment example given in Section 3 we will illustrate how it may be more complicated than this. More precisely, ϕ_T:

(a) will be a high dimensional vector;

(b) will be a vector whose length changes with the time index.

In practice "effects" are often observed before "causes" in a causal network which makes the raw causal network an unsuitable framework for propagating new information round the system. It is helpful to define a framework for absorption of information which is non—directional. When a causal network is not dynamic the most common such framework for probability propagation is the *junction tree* [1,9,10,13] which is derived from a causal network as follows.

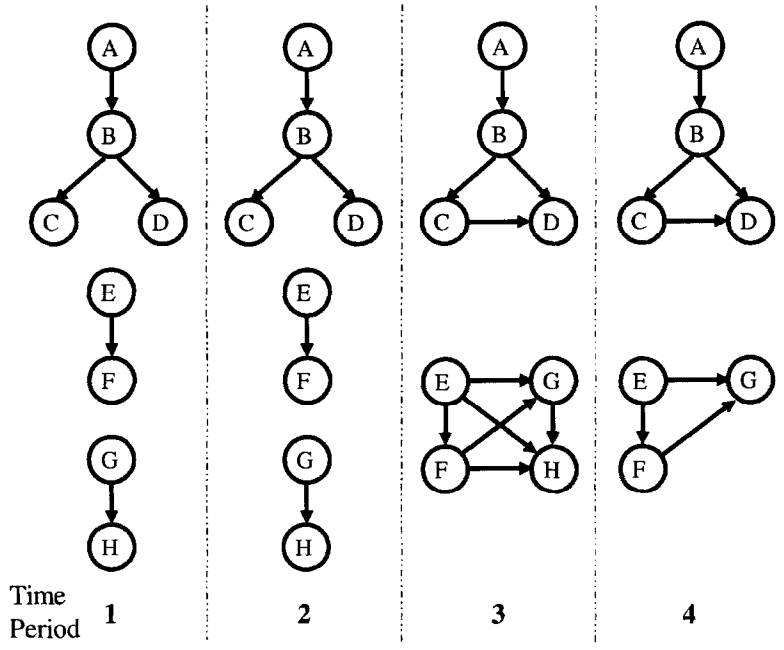

Fig. 4. Margins of states where the distinction of ω_t is chosen to preserve the shape of the causal network in time given all data collected strictly before the time period index. The data collected is assumed to be measurements of brand sales of aggregates given in the text plus some independent errors.

First, new directed edges joining unconnected parents of each node in the causal network ensure that no cycles are created producing a new causal network. This process is repeated until the set of parents in each node of the causal network are joined by an edge. The resulting graph is called decomposable. In the causal network of Fig. 3 the directed edges which need to be added are: $(B[1], A[2])$, $(B[2], A[3])$, $(B[3], A[4])$, and between all nodes not already connected in each of the sets $\Xi[i] = (E[i], F[i], G[i], H[i], E[i+1], F[i+1], G[i+1], H[i+1])$, $i = 1, 2$, and the set $\Xi[3] = (E[3], F[3], G[3], H[3], E[4], F[4], G[4])$. Note here that many edges need to be added. On the other hand each of the four causal networks of Fig. 4 used in our predictive dynamic probability propagation have all parents already connected and so do not need any such modification.

The *cliques* of a decomposable graph are defined to be its maximally connected subsets. For example, the causal network of Fig. 2 has 5 cliques (A, B), (B, C), (B, D), (E, F) and (G, H). In the third causal network they are (A, B), (B, C, D), (E, F, G, H). The cliques $c[i]$, $1 \leqslant i \leqslant m$, of a decomposable causal network can always be ordered so that they satisfy, the running intersection property [14,32], i.e., for all i, $2 \leqslant i \leqslant m$, there is an index $r(i)$, $1 \leqslant r(i) < i$ such that:

$$\text{if } d[i] = c[i] \cap \bigcup_{j=1}^{i-1} c[j] \text{ then } d[i] \subset c[i] \cap c[r(i)].$$

Henceforth, we shall label the vector of variables contained in a clique $c[i]$ by $\phi[i]$, $1 \leqslant i \leqslant m$. A *junction forest* of a causal network has as nodes the cliques of that network numbered consistently with the running intersection property and with clique $c[j]$ connected by an undirected edge to clique $c[i]$, $j < i$, if and only if $j = r(i)$ and $d[i]$ is not empty, where $r(i)$ and $d[i]$ are defined above. A *junction tree* is just a connected subgraph of a junction forest.

Here we will call $d[i]$ the separator of the cliques $c[i]$ and $c[r(i)]$ and denote by $\phi_1[i]$ the vector of variables it contains. It is straightforward to check that

$$c[i] \coprod c[1], \ldots, c[m] \mid d[i], \quad 2 \leqslant i \leqslant m. \tag{2.6}$$

Let $p_i(\phi[i])$, $\{q_i(\phi_1[i])\}$ denote the marginal densities of $\phi[i]$, $\{\phi_1[i]\}$ respectively where, if $d[i]$ is empty or $i = 1$, $q_i = 1$, $1 \leqslant i \leqslant m$. Then because of the conditional independence relations given above it can be shown that the joint density $p(\theta)$ of the original variables can be written

$$p(\theta) = \prod_{i=1}^{m} \frac{p_i(\phi[i])}{q_i(\phi_1[i])}, \tag{2.7}$$

where p is the marginal density of the subvector $\phi[i]$ of $c[i]$ and q is the marginal density of the components of $\phi_1[i]$ in $d[i]$ defined above (or as equal to 1 if $d[i]$ is empty). Note that two causal networks with the same undirected version will give rise to the same breakdown given above and so in this sense the decomposition is not dependent on the causal order of the variables.

The fact that the junction tree has a structure which is not dependent on the causal ordering of the system makes it ideal as a framework for driving algorithms for the quick absorption of information into the system. In the current literature it is usually assumed that a subset of states are observed without error. The algorithms then update the usually discrete densities of the remaining states in the light of this [9,33]. Here we are in a slightly different setting for two reasons: first that the distribution of states is typically continuous and second that we observe only a function of states with error. The first difference causes no problems in principle since the propagation algorithms are equally valid in a continuous setting although there are computational issues involved once we drift outside the Gaussian family (see, e.g., [13]). The second difference is much more significant because the learning algorithm described above may no longer be invalid. However, in the particular case when errors are independent and each observation is a function only of states in a single clique it is straightforward to show that this not so. To show this we just create one new clique $c^*[i]$ for each observation $y[i]$ which contains that observable together with those states of which it is a function. Since $c^*[j]$ only contains states in a single clique $c[j]$ (say) the conditional independence implicit in our observation equation tells us that if we connect $c^*[j]$ by an undirected edge to $c[j]$ for each $y[i]$ in a new junction tree J^* whose edges between states follows the original tree J, then J^* is also valid. The conventional propagation algorithms (e.g., [9]) used on J^* instead of J now apply directly. Once the propagation has been completed, the cliques together with their connecting edges, in J^* and not J can be removed without loss since they contain no unknown quantities not contained in other cliques once the observations are seen—for a more detailed discussion of this procedure see [5].

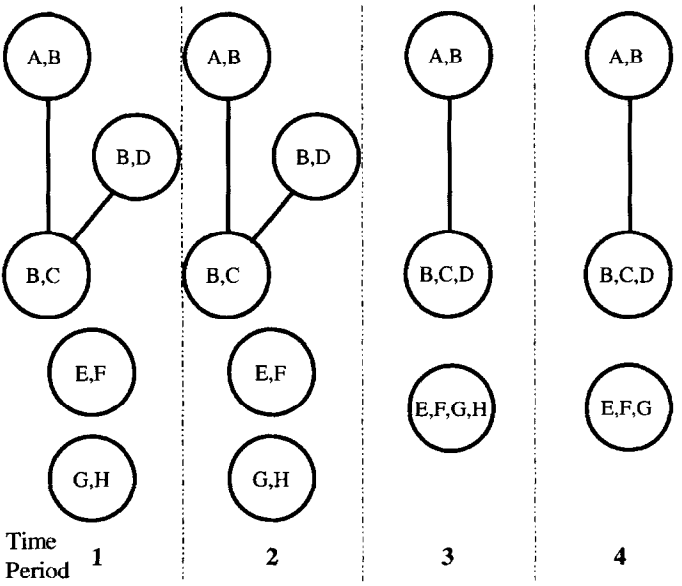

Fig. 5. Marginal junction forests associated with the causal networks of Fig. 4.

Henceforth, we will assume that an algorithm like the one described above will be employed to accommodate all information collected in a given time period. In our marketing example the sequence of valid junction forests corresponding to the causal networks in Fig. 4 are given in Fig. 5. Notice that in time period 3 the upper junction tree has been adapted so that an observation of $C + D$ has states which all lie in the same clique. Formal procedures for making efficient adjustments to the tree of this type are given in the body of the paper.

In many examples both the states and the observation will be Gaussian. In this case the updating of the marginal densities of the cliques in the light of an observation Y and subsequent adaptation (called "absorb" by Jensen) is extremely simple. Thus, consider the adaptation on the junction tree J^* derived from the original tree J above. The joint density of the states in $c^*[j]$ given Y will be Gaussian (see [16] for the appropriate equations). These states will form the separator of $c^*[j]$ from $c[j]$ in J^*. The distribution of cliques in the junction tree containing $c[j]$ will then be updated using the following algorithm sequentially. Once the distribution on all the cliques has been updated the clique $c^*[j]$ can be removed without loss to give us back the original junction tree J as discussed above. The distribution of variables in the cliques of other trees in the junction forest will remain unchanged.

Let the vector of random variables in $c[i]$ be $\phi[i] = (\phi_1[i], \phi_2[i])$ and before Y is accommodated into this clique let $\mu_j = E(\phi_j[i])$, $\Sigma_{jj} = \mathrm{Var}(\phi_j[i])$, $j = 1, 2$, and $\Sigma_{12} = \mathrm{Cov}(\phi_1[i], \phi_2[i])$, where $\phi_1[i]$ is the vector of the newly updated separator. Then $\mu_1^* = E(\phi_1[i] \mid Y)$ is the updated mean of this separator and $\Sigma_{11}^* = \mathrm{Var}(\phi_1[i] \mid Y)$ is its updated covariance matrix which can be read directly from the adjacent updated clique. Since the distribution of $\phi_2[i] \mid \phi_1[i]$ remains fixed under the updating, the usual multivariate normal

theory (see, e.g., [16]) now gives us that $\mu_2^* = E(\phi_2[i] \mid Y)$, $\Sigma_{22}^* = \mathrm{Var}(\phi_2[i] \mid Y)$ and $\Sigma_{12}^* = \mathrm{Cov}(\phi_1[i], \phi_2[i] \mid Y)$ are given by the equations:

$$\mu_2^* = \mu_2 + A(\mu_1^* - \mu_1),$$

$$\Sigma_{21}^* = A\Sigma_{11}^*,$$

$$\Sigma_{22}^* = \Sigma_{22} - A[\Sigma_{11} - \Sigma_{11}^*]A^{\mathrm{T}},$$

where $A = \Sigma_{21}\Sigma_{11}^{-1}$. These results are discussed in detail in [29]. Finally, to use these equations to pass information to new cliques we need to calculate the distribution of the new separators contained in $c[i]$. These will be subvectors of $c[i]$. So in the Gaussian case this step is trivial: we just need to pick the correct components from the new mean vector and covariance matrix of $\phi[i]$. Interestingly, in non-Gaussian continuous cases—unless there are appropriate properties on the clique margins—as exists for hyper Dirichlet and hyper Wishart families, for example, see [3]—this marginalising step may be very time consuming involving the calculation of an integral. Of course no such problem exists in the discrete case since the analogous operation is simply a summation. On the other hand the updating step analogous to the one given above, is often simple—being the algebra of multiplying two functions together.

3. States and the efficient representation of dynamic graphs

There are usually many representations of a probabilistic structure in terms of a graph and it is critical that the graphical representation which guides the probability propagation at each time point does so as efficiently as possible. Which graphical representation is most computationally efficient is determined by many considerations including the predictive capability required, the type timing and quantity of relevant updating information and the technological method which is currently available to process this information. In this paper we will assume that we will use a junction tree algorithm like the ones cited in the last section to process the data.

To achieve this efficiency and also to make a representation more transparent it is necessary to define the appropriate sequence of random vectors, indexed in time, whose joint distribution can be represented on a junction tree, such that the distribution of a random vector at time t contains all information required to calculate the probability statements that might be required for future predictions after time. Thus we need:

(a) A sequence of state vectors, indexed by time.
(b) A junction tree which stores the distribution of each state vector at any time t.
(c) Algorithms which adjust this distribution—i.e., adjust the cliques junction tree and associated distributions on clique margins—in the light of incoming data at that time point t. One such algorithm was described in the previous section.
(d) An algorithm which takes the cliques, junction tree and associated distributions on clique margins on the state vector at time $t-1$ and transforms these to a class of cliques, a junction tree and new associated distributions on clique margins at time t, $t = 1, 2, 3, \ldots$. This process was illustrated in our example given above.

Sometimes exact algorithms give rise to distributions on states whose joint distributions have no compact graphical representation, i.e., they cannot be represented by a sparse graph whose associated cliques have a small number of states. In this case it is often expedient to substitute fast approximate algorithms for slow exact ones. Before we formalise this procedure we give another example, this time an environmental model where the relevant prediction junction trees evolve in time in a much more structured way.

Example 3.1. Pentificating Puffs from a steady release.

This is a process for predicting the spread of gaseous waste after an accident [28] using junction tree propagation algorithms (see [29] for more details). Briefly, after a nuclear accident puffs containing a mass of dangerous radioactivity are emitted from a building. Each puff is transported through the atmosphere by a wind field and as it is transported it becomes larger and the mass it contains becomes more spread out. Once a puff grows to a certain size at a time determined by the atmospheric conditions, the dispersion model allows it to fragment into five pieces so that different sectors of the large puff can be transported by the wind field depending on where each sector lies. Fragmentation of fragments will eventually occur and so on. Each fragment is determined by a complicated but deterministic formula which is a function of many known environmental covariates associated with the release. Readings $Y(t,s)$ of air concentration at time t and site s are taken periodically. Physics tells us that these concentrations are a linear combination of mass fragments which exist at time t and have part of their puff over the site s—larger puffs contributing less of their mass to this reading together with an independent observational error.

This process, defined by physicists, is Markov in the sense that at time T the joint distribution of future mass fragments—and hence the distribution of all future observations—can be expressed as a function of the joint distribution of mass fragments which exist at time T. It follows that one valid state space at time T, which will be sufficient to provide any distribution over observations and states at that time, will be the vector of mass fragments/puffs existing at or before time T.

Unfortunately for large T this is typically an extremely long vector. However, we show in [29] that, provided puffs are released as a Markov process and all observations are linear combinations of puff masses which are fragments of the same "parent" puff, the joint distribution on this vector can be stored on a junction forest whose cliques are either:

(a) masses of puffs adjacent in time emitted on or before time T (a random vector of length 2); or
(b) masses of a "parent" puff and its associated fragments (often either 5 or a random vector of length 6).

Cliques are linked by an edge in the junction forest if and only if they have a component in common in their random vector. This component is always either a parent fragment or a puff emitted at source. Because cliques are of small dimension, this is a very efficient way of storing the joint distributions on states. An example of such a tree is given in Fig. 6. If emissions are independent at the source then no puffs of the form (i) need be included in the state vector and the forest, instead of being a single tree, becomes a forest with one tree for each puff emitted at or before time T.

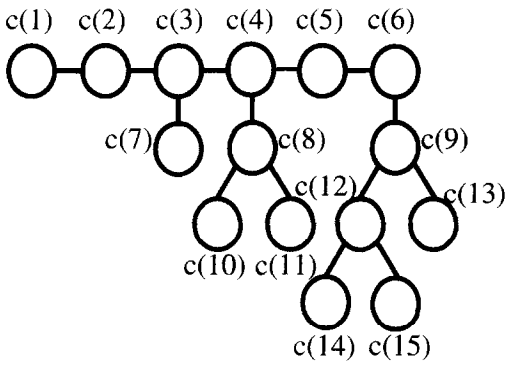

Fig. 6. A Puff model junction tree.

In Fig. 6, for simplicity, puffs are depicted splitting into 2 rather than 5. $c(1), \ldots, c(6)$ are cliques associated with the first 7 emissions—containing adjacent pairs. Other cliques are numbered in the order they appeared. Cliques $c(7)$, $c(8)$ and $c(9)$ are parents and fragments of previously undivided puffs. Cliques $c(10)$, $c(11)$, $c(12)$ and $c(13)$ are parents and fragments of fragments and so on.

Now suppose at time T we take an observation which is a linear combination, with error, of puff fragments which are all components of a single clique vector given above. Then the method for accommodating information in a Gaussian network outlined in Section 2 is valid, the cliques and junction tree of the state vector remain unchanged and the joint distributions on the vectors in the cliques are updated as we described. However, if an observation Y is a linear combination of components for which this is not so then, posterior to observing Y, dependencies implicitly denied by the original junction forest will be induced. In this case the forest will need to be modified before data can be accommodated. In Section 5 we describe a way of doing this.

In this application it is important to note that the system need only determine what happened at the source and predict future local contamination—a linear combination of future mass fragments and the source emissions. Therefore, only a subset of states in the process are essential for determining these distributions—the source masses and the mass fragments terminal to the forest. The other states are used only to efficiently transmit information around the forest. In later sections we will discuss how a junction tree can be automatically speeded up by simplifying or removing completely some of these transmitter cliques.

4. A formalisation

A state vector ϕ_T at time T is a random vector whose distribution at time T given the data D received up to and including time T is sufficient, when used together with distributional information known a priori, to determine:

(a) The expected utility of the optimal policies for any utility function which might be called at time T.

(b) The expected utility of the optimal policies for any utility function U which might be called at time $T^1 > T$ as a function of covariates whose values are certain to be known at time T^1.

We saw in the previous section that a state vector is not uniquely defined but that some representations allowed much better computational efficiency than others. We also saw that the appropriateness of a state vector depended heavily on the questions needing to be answered by the system—i.e., the utilities under which policies might be chosen. For example, for the dynamic linear model of Section 2 in order to estimate $\{\theta_t\colon t = 1, 2, 3, \ldots\}$ the state vector needs to store past distributions of θ_t in some way—e.g., by setting $\phi_T = \{\theta_1, \ldots, \theta_T\}$. On the other hand, if questions asked only involved the future after time T then it was sufficient to let the state vector at time T be θ_T itself.

A state vector ϕ_T is not necessarily minimally sufficient for determining expected utilities in (a) and (b) since some of the components of ϕ_T may be included to facilitate quick propagation using the L & S algorithm (Lauritzen and Spiegelhalter [14]). It will be useful later to write $\phi_T = (\phi_T(1), \phi_T(2))$, where $\phi_T(1)$—called *essential states*—are a minimal set of components of ϕ_T which on their own satisfy requirements (a) and (b) and where the rest $\phi_T(2)$ will be called *transmission states*.

A *state process* is a time sequence of state vectors. Note that there is always a model which sets states $\theta_t = (\theta_t, \acute{Y}_t)$, $t = 2, 3, \ldots$, where $\acute{Y}_t$ are all the random variables observed in the time interval $(t-1, t]$, which makes $\phi_t\colon t = 1, 2, \ldots$, a state process although such a representation will usually be inefficient. Each state vector at time T in a state process will have associated with it a triple (T_T, C_T, S_T) consisting of, respectively, a junction tree, a set of cliques (labeling the nodes of the tree) and a set of separators (labeling the edges of the tree). It will also have a set of marginal probability distributions on cliques—i.e., a set $P_T = \{p_i(\cdot)\colon c[i] \in C_T\}$. Henceforth call $\{T_t, C_t, S_t, P_t; t = 1, 2, 3, \ldots\}$ a *junction tree process* associated with the state process $\{\phi_t\colon t = 1, 2, 3, \ldots\}$. A *junction tree state* J_T at time T is simply $J_T = (T_T, C_T, S_T, P_T)$, $T = 1, 2, 3, \ldots$. A *transmitter clique* is a clique whose components are entirely transmitter states, otherwise it is called an *essential clique*.

When all the densities in P_t lie in a parameterised family $t = 1, 2, 3, \ldots$ then their storage will usually be in terms of the values of these parameters. For example, in the case of a Gaussian process, $p_i(\cdot) \in P_t$ can be stored in terms of their mean vectors μ_i and covariance matrices Σ_i.

Because of the Markov structure of the state process at each time point $t = 1, 2, 3, \ldots$ it is convenient to specify a *junction tree equation*, i.e., an equation which, for each t, specifies the junction tree state J_t as a function of J_{t-1} and any observations in the time interval $(t-1, t]$. Usually transformations are a composite of one of three different types:

(1) Transformations corresponding to descriptive state equations. In Example 3.1, two types of transformation of this kind were described: we will use one of these for illustration. When a new puff of mass Q_T is emitted from the chimney at time T, it forms a new clique $c = (Q_T, Q_{T-1})$ so that $C_T = C_{T-1} \cup \{c\}$. The previous puff mass Q_{T-1} lies in the clique $c' = (Q_{T-1}, Q_{T-2})$ in C_{T-1}. Set $S_T = S_{T-1} \cup \{Q_{T-1}\}$. The tree T_T is formed from the junction tree T_{T-1} by adding a node labeled c and an edge joining c to c' labeled Q_{T-1}. The set of densities P_T just adds the density of the clique c to P_T. The clique margin of c is calculated by multiplying the density of Q_{T-1}—calculated from the clique margin c'—by the

conditional density of $Q_T \mid Q_{T-1}$, the latter being known a priori. After such a transformation, states which were essential at time $t-1$ may well become just transmission states at time t. Indeed in Section 6 we will illustrate how they can become completely redundant.

(2) Transformations induced by the arrival of observations. In Section 2 we cited a method of updating a junction tree process in the light of an observation which was a function of states all lying in a single clique. This transformation was particularly simple because it set $(T_T, C_T, S_T) = (T_{T-1}, C_{T-1}, S_{T-1})$ and changed only P_{T-1} to a new set of densities P_T in the light of the observation. Again this can make some states in T_{T-1} redundant at time T—for example, when such states acted as a surrogate for data that is now observed.

(3) Transformations which redefine a junction tree state to make it more compact and so make quick algorithms feasible. Exact versions of these are discussed in Section 6 and approximate methods described in Section 7.

Because transformations of the type (1) tend to be very specific to an application, in this paper we will concentrate our attention on the second two. Examples of the first type are discussed in detail by Gargoum and Smith [6] and Gargoum [5].

5. Transforming junction trees to assimilate data

In Sections 1 and 2 we gave examples where, because an observation was a function of states in different cliques $\{c[1], c[2], \ldots, c[r]\} = D$ the usual junction tree algorithms for probability propagation were no longer applicable. In this section we describe an algorithm which modifies the junction tree state so that the new clique and junction forest structure can legitimately use the usual junction tree propagation algorithms. Such a forest will contain a clique with all the variables in D defined above.

Let $h[b, c; T]$ denote a path between nodes b and c in C of (T, C, S, P) such that $h[b, c; T]$ has no repeated nodes. Note that if $h[b, c; T]$ exists it is unique because then both b and c lie in the same tree of the forest. In the junction forest of the period 2 of Fig. 5, we observe the aggregate sales of $C + D$, with error, and we note that in the junction forest there is no one clique which contains brands C and D. However, there is a path between the two cliques $b = \{B, C\}$ and $c = \{B, D\}$ with no repeats—namely the edge joining b to c.

Let $T(c[1], c[2], \ldots, c[r])$, $c[1]$, $c[2], \ldots, c[r]$ in C denote the smallest subforest containing each of the paths $h[c[i], c[j]; T]$, $1 \leqslant i, j \leqslant r$. Let $T_j(c[1], c[2], \ldots c[r])$, $1 \leqslant j \leqslant k$, (or briefly T_j) denote the k disconnected trees which comprise the forest $T(c[1], c[2], \ldots, c[r])$ and $B[j]$ denote the nodes of T_j, $1 \leqslant j \leqslant k$. Let $D[j] = D \cap B[j]$ so that $D[j]$ are disjoint, $1 \leqslant j \leqslant k$, and their union is D. We now further partition $B[j]$ into $C[i, j]$, where $C[i, j] = \{b \in B[j]\colon \min_{c \in P_j} \lambda(h[b, c; T_j]) = i\}$, $i = 1, \ldots, l_j$, where $\lambda(h)$ is the number of nodes between b and c in the path h and l_j is the maximum distance between a node in $B[j]$ and those in $D[j]$.

Let $c[i, j] = \bigcup_{c(i) C[i,j]} c(i)$, $1 \leqslant i \leqslant l_j$, $1 \leqslant j \leqslant k$, and let

$$C^{\#}[j] = \{c[i, j]\colon 1 \leqslant i \leqslant l_j\}, \qquad C^{\#} = \bigcup_{1 \leqslant j \leqslant k} C^{\#}[j].$$

Let T' be a forest whose nodes are $C' = \{C \setminus D\} \cup C^{\#} \cup \{d\}$, where $d = \bigcup_{i=1}^{r} c[i]$ such that:

(i) Two nodes b, c in $C \setminus D$ are connected by an edge in T' iff they are connected by an edge in T.

(ii) A node $b \in C \setminus D$ is connected to a node $c[i, j] \in C^{\#}$ iff $C[i, j]$ contains a node to which b was connected in T.

(iii) The node $c[i, j] \in C^{\#}$ is connected to the node $c[i', j'] \in C^{\#}$ iff $j = j'$ and $i' = i - 1$ or $i + 1$.

(iv) The node d is connected to a node $b \in C'$ by an edge iff $b = c[1, j]$ for some j, $1 \leqslant j \leqslant k$.

It is straightforward to check that T' is a forest. Since both the cliques C' and the separators S' of T' are defined above, it remains to define the clique marginal probability distributions P'.

Each clique in $C \setminus D$ is given the same margin in C' as it had in C. The margin of $c[i, j]$ is calculated by marginalising appropriately the joint density of states in T_j in (T, C, S, P). Finally the clique d has margin obtainable from taking the product of the margins $c[0, j]$, $1 < j < k$, calculated above. It is straightforward to check that performing this construction on the junction forest T of time period 2 of Fig. 5 gives the junction forest T' of the period 3 of this figure.

Now we have the following theorem.

Theorem 5.1. *If the junction tree state (T, C, S, P) is valid then so is the junction tree state (T', C', S', P') defined above.*

Proof. This is a direct consequence of Pearl's [17] d-separation Theorem—see Appendix A. □

Having transformed T to T' we note that all the observed variables lie in the one clique d. It is now valid to use the updating algorithm described in Section 2 to recalculate the new marginal probabilities on the states of P'.

Note that the joint density $p^{(j)}(\phi)$ of the variables lying in the cliques in $B[j]$ is given by formula (2.7) so

$$p^{(j)}(\phi) = \frac{\prod_{c[i] \in B[j]} p_i(\phi)}{\prod_{s[i,i'] \in I[j]} q_{ii'}(\phi)}, \tag{5.1}$$

where $p_i(\phi)$ is the margin of $c[i]$, $q_{ii'}(\phi)$ is the density of the separator $s[i, i']$—associated with an edge in T_j linking $c[i]$ and $c[i']$ where $I[j]$ is the set of all separators in T_j.

In the special case when all variables in cliques in $B[j]$ are jointly Gaussian, the mean and covariance matrix of these variables are easily calculated. The mean vector can be read directly from the means of the densities $p_i(\phi)$. The covariance matrix can be obtained by first numbering the cliques in $B[j]$ *regularly*: i.e., so that a junction graph whose undirected version would be T_j with cliques numbered compatibly would be valid. Now the covariance matrix is constructed inductively using this numbering.

Thus let $\phi(1)$ denote all variables whose covariance matrix has already been calculated (at first the variables in a single clique) and let $\phi(2) = (\phi_1(2), \phi(s))$ be a vector of variables

in a clique adjacent in T_j to those already having their joint covariance matrix calculated, where $\phi(1) = (\phi_1(1), \phi(s))$ so that $\phi(s)$ is the random vector of variables common to both $\phi(1)$ and $\phi(2)$, and $\phi_1(j)$ is the residual of $\phi(j)$, $j = 1, 2$. If the covariance matrix of $\phi(j)$ is given by $\Sigma[j]$:

$$\Sigma[j] = \begin{pmatrix} \Sigma_1[j] & \lambda_j(s) \\ \lambda_j^{\mathrm{T}}(s) & \Sigma[s] \end{pmatrix},$$

where $\Sigma_1(j)$ is the covariance matrix of $\phi_1(j)$ and $\Sigma[s]$ is the covariance matrix of $\phi(s)$, then the covariance matrix of $\phi^*(1) = (\phi_1(1), \phi_1(2), \phi_1(3))$ is given by

$$\Sigma^*[1] = \begin{pmatrix} \Sigma_1[1] & \lambda(1,2) & \lambda_1(s) \\ \lambda^{\mathrm{T}}(1,2) & \Sigma_1[2] & \lambda_2(s) \\ \lambda_1^{\mathrm{T}}(s) & \lambda_2^{\mathrm{T}}(s) & \Sigma[s] \end{pmatrix},$$

where $\lambda(1,2) = A_1 \Sigma(s) A_2^{\mathrm{T}}$ and $A_j = \lambda_j(s)\Sigma^{-1}(s)$, $j = 1, 2$, using standard multivariate normal theory (e.g., [16]). This completes the inductive step since we now replace $\phi(1)$ by $\phi^*(1)$ and repeat the procedure, and keep doing this until the covariance matrix of all variables in cliques contained in $B[j]$ have been calculated. It is now a trivial matter to read the mean and covariance matrices of $c[i, j]$ from the mean and covariance matrix of the vector calculated above—and hence to calculate all the new densities lying in P'.

In the non-Gaussian case the calculation of the density of $c[i, j]$ may need an integration or a summation which will slow the process down dramatically. However, with some loss of efficiency we can replace T' with a different junction tree having nodes which replace C' with $\{c'\}$ where

$$c' = \bigcup_{\substack{1 \leqslant i \leqslant l_j \\ 1 \leqslant j \leqslant k}} c[i, j].$$

Substituting $\{c'\}$ for C' in the constructions (i)–(iv) gives a valid junction tree representation.

Finally note that the use of (5.1) without integration requires that, as for the updating algorithm described in Section 2, the margins $q_{ii'}(\phi)$ can be retrieved in closed form from the clique margins—but the algorithm defined in this section requires no further conditions to work quickly and so can be seen as a generalisation of the one given in Section 2.

We finish this section with an example.

Example 5.1. Deriving a new junction tree for probability propagation.

You take an observation Y_t whose distribution depends explicitly on the state vector ϕ_t only through those components of ϕ_t which lie in cliques $D = \{c[3], c[7], c[12], c[15], c[23], c[24]\}$. To draw a forest which is both valid and has all these components lying in a single clique d, first construct the trees T_1 and T_2 which are defined as the union of all paths between the cliques in D in T, the tree T given in Fig. 7. These two subtrees are given in Fig. 8. Using Eq. (5.1) we therefore see that the joint densities $p^{(j)}(\cdot)$ of states in T_j, $j = 1, 2$, are given by

$$p^{(1)}(\cdot) = \frac{p_1(\cdot)p_2(\cdot)p_3(\cdot)p_4(\cdot)p_6(\cdot)p_7(\cdot)p_9(\cdot)p_{12}(\cdot)p_{13}(\cdot)p_{14}(\cdot)p_{15}(\cdot)}{q_{1,2}(\cdot)q_{2,3}(\cdot)q_{1,4}(\cdot)q_{4,6}(\cdot)q_{6,7}(\cdot)q_{4,9}(\cdot)q_{9,12}(\cdot)q_{12,13}(\cdot)q_{13,14}(\cdot)q_{14,15}(\cdot)}$$

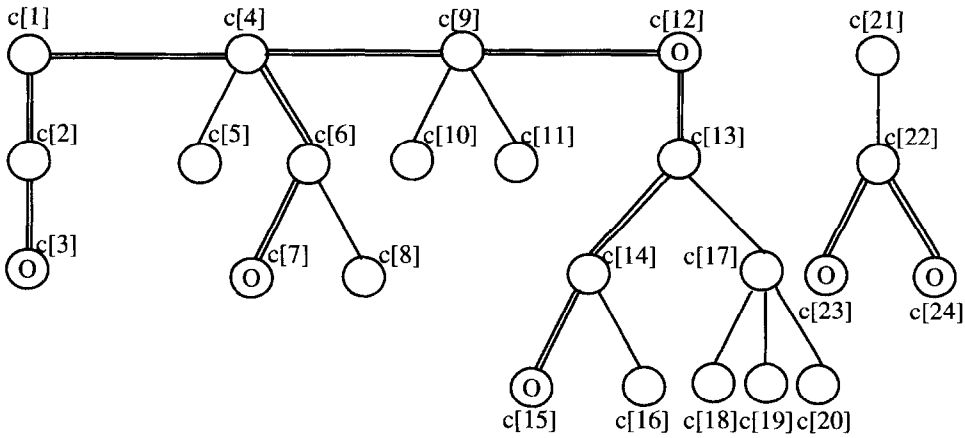

Fig. 7. A junction forest before a function of variables in the set of cliques $D = \{c[3], c[7], c[12], c[15], c[23], c[24]\}$ is observed.

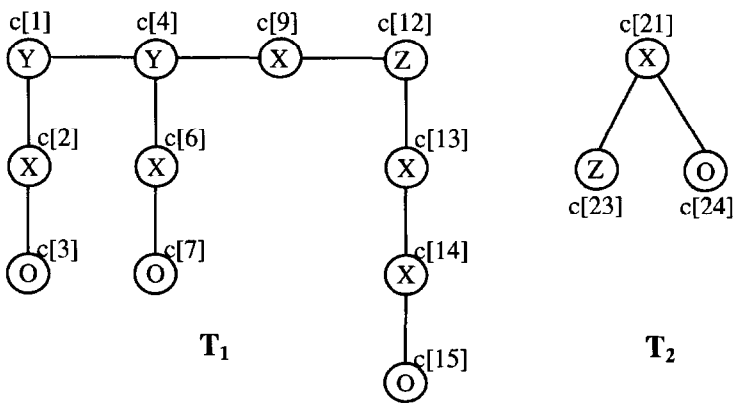

Fig. 8. The constructed subforest $\{T_1, T_2\}$ with $D[1] = \{c[3], c[7], c[12], c[15]\}$ and $D[2] = \{c[23], c[24]\}$. We now combine cliques in each tree which have the same marking—i.e., are the same distance from the set of observed nodes.

and

$$p^{(2)}(\cdot) = \frac{p_{21}(\cdot)p_{23}(\cdot)p_{24}(\cdot)}{q_{21,23}(\cdot)q_{21,24}(\cdot)}.$$

From these joint densities it is possible to calculate the clique margins of the newly created cliques $c[0, 1], c[1, 1], c[2, 1], c[0, 2], c[1, 2]$ (see Fig. 10). The algorithm now reconstructs the tree as depicted in Fig. 9.

Interestingly it is easy to check that $c[2, 1]$ and $c[1, 2]$ can be calculated avoiding an integration step. Thus, the clique margin of $c[1, 2]$ is just the clique margin of $c[21]$ and the clique margin of $c[0, 2]$ is given by

$$\frac{p_{21}(\cdot)p_{24}(\cdot)}{q_{21,23}(\cdot)q_{21,24}(\cdot)}.$$

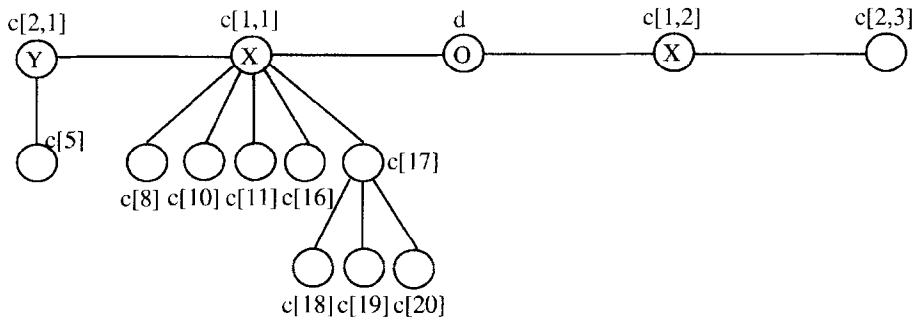

Fig. 9. The reconstruction of T to T', completed. Here $d = c[0,1] \cup c[0,2]$.

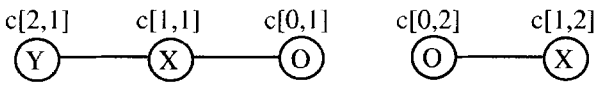

Fig. 10. The transformation of $\{T_1, T_2\}$ ready for the construction of T'. $c[0,1] = c[3] \cup c[7] \cup c[12] \cup c[15]$, $c[1,1] = c[2] \cup c[6] \cup c[13] \cup c[14]$, $c[2,1] = c[1] \cup c[4]$, $c[0,2] = c[23] \cup c[24]$, $c[1,2] = c[21]$.

The density of the separator of $c[0,2]$ and $c[1,2]$ which is the union of the separators of $c[21]$ and $c[23]$ and $c[21]$ and $c[24]$ in the original junction tree is given by

$$\frac{q_{21,23}(\cdot)q_{21,24}(\cdot)}{r_{21}(\cdot)},$$

where $r_{21}(\cdot)$ is the density of variables lying in both separators, and equal to one if this set is empty. However, this is not so for $B[1]$, so if states in this tree were not Gaussian it may be necessary to place all these nodes in a single clique. It has been our experience in the nuclear coding that in fact this construction combines cliques on trees T_j which are small—indeed for many parameter settings we just combine cliques with the same parent or cliques from different trees in the forest.

6. Other exact algorithms

6.1. Splitting up trees

It will have been noted that if a forest T is split up into many trees then the propagation algorithm described in Section 2 tends to work more quickly. The Chop algorithm described below transforms a junction tree state $J = (T, C, S, P)$ at a given time T to another state $J^* = (T^*, C^*, S^*, P^*)$ whose forest has more trees. It can be applied whenever a separator $s \in S$ becomes degenerate—for example, by being observed directly and without error.

If a separator $s_{12} \in S$ between cliques c_1 and $c_2 \in C$ becomes degenerate then clearly c_1 and c_2 are now independent of one another. The following transformation of (T, C, S, P) is therefore valid.

The Chop transformation sets

$$S^* = S \backslash \{S_{12}\},$$
$$C^* = \left(C \backslash \{c_1, c_2\}\right) \cup \left(C_1^*, C_2^*\right),$$

where c_i^* has as its components the components of c_i less the components of s_{12}, $i = 1, 2$. T^* is the forest with the new clique and edge set C^* and S^*, respectively—so the tree $T_{12} \in T$ containing s_{12} is replaced by two trees—one containing c_1 and the other containing c_2.

As time passes inevitably the number of cliques multiplies unless the states of the process are defined carefully. This can be done, to some extent, automatically. The next transformation loses irrelevant transmitter cliques. It is in fact just a version of Shachter's [23] Barren node reduction algorithm for decision influence diagrams but stated for junction trees in a dynamic setting.

The Chop transformation takes the junction tree states $J = (T, C, S, P)$ to the state $J^* = (T^*, C^*, S^*, P^*)$ where $c^* \in C^* \leqslant C$, iff c^* is an essential clique or if c^* labels a node in T which lies on a path in T between two essential cliques of T. Separator $s^* \in S^* \leqslant S$ iff s^* is an edge lying in a path between two essential cliques in (T, C, S, P). The forest T^* is the subforest of T whose nodes are C^* and edges S^*, where C^* and S^* are defined above. For $c^* \in C^* \leqslant C$ we set $p^*(c) = p(c)$ where $p(c) \in P$.

Theorem 6.1. *If J is a junction tree state of a process at time T then so is J^* defined above.*

Proof. See Appendix B. □

If $c^* \in C^*$ then its margin will just be that given in P. If $c^* \in C^*$ is not in C then its margin is just the margin of the subvector of c in C containing it—which will be the conditional density of c_i given s_{12}, where $i = 1$ or 2 since s_{12} has a degenerate distribution. Implicitly Spiegelhalter and Lauritzen [30] use this property for discrete problems, calling it the global independence property. Queen and Smith [21] use it implicitly for some continuous time series.

6.2. *Keeping transmitter cliques small*

It was seen in Section 5 that adjusting junction tree states to allow accommodation of information from data has a tendency to increase the clique size with a consequent loss of efficiency. A junction tree transformation call *Thin* keeps the size of transmitter cliques as small as possible by replacing the transmitter clique by the union of its separators. Thus the transformation Thin takes the junction tree state $J = (T, C, S, P)$ to the junction tree state $J^* = (T^*, C^*, S^*, P^*)$ as follows:

C^* consists of the essential cliques of C and replaces each transmitter clique $c \in C$ by c^*, where c^* is the subvector of c consisting of the components of c which are contained in at least one of its separators—corresponding to edges in T connecting to the node in T corresponding to c. Set $T^* = T$ and $S^* = S$. If $c^* \in C \wedge C^*$ its density is unchanged and if $c^* \in C^* \backslash C$ then the density of c^* is the appropriate margin of $p(c)$ where $c \in C$ contains

the c^* subvector. Notice then in the Gaussian case, the set of new clique margins is trivial to calculate, being given by the sub mean vector and appropriate sub covariance matrix of the containing clique in C.

As the process evolves it may happen that transmitter cliques may be removed expediently even if they lie in a path of a junction tree between two essential cliques. In $J = (T, C, S, P)$ let $n(c[j]) \subseteq C$, called the *neighbours* of $c[j]$ in C, be the set of cliques connected to $c[j]$ in T by an edge. Let $s[i, j] \in S$ be the edge which connects $c[i]$ to $c[j]$ in T and let $\#s[i, j]$ denote the dimension of the vector of states $s[i, j]$.

6.3. *Combining transmitter cliques*

The final transformation in this section is called *Contract*. It acts on two transmitter cliques $c[j]$ and $c[j'] \in C$ which are adjacent in T. It takes (T, C, S, P) to (T^*, C^*, S^*, P^*) where:

$$C^* = \left(C \backslash \{c[j], c[j']\}\right) \cup \{c^*[j, j']\},$$

$$S^* = \left(S \backslash \{s[j, j']\}\right) \cup \{s^*[j, j']\},$$

where $s[j, j']$ denotes the set of all edges in S which are adjacent to either $c[j]$ or $c[j']$ in T and $s^*[j, j']$ is a set of edges from nodes in C^* (and C) which are adjacent to $c[j]$ or $c[j']$ in C connecting each such node to the new node $c^*[j, j']$. Let T^* be the graph whose nodes and edge set is, respectively, C^* and S^* defined above. Let the density of $c^* \in C^*$, $c \neq c^*[j, j']$, in P^* be identical to the density of c in P. Define the density of $c^*[j, j']$ by

$$\frac{p_j(\cdot)p_{j'}(\cdot)}{q_{jj'}(\cdot)}.$$

where p_j and $p_{j'}$ in P are the densities of $c[j]$ and $c[j']$, respectively, and $q_{j'}$ is the density of $s[j, j']$, their separator in T. Note that when the variables in $c^*[j, j']$ are Gaussian, its mean and covariance matrix can be obtained using the formula (5.1).

The Contract transformation works with Thin to shorten the length of paths between essential cliques when it is computationally efficient to do this. Computational efficiency obviously depends quite critically both on the algorithms used by the system and the machine doing the computing. But larger cliques and trees with more edges will almost inevitably lead to a loss of efficiency. There are many candidates which might determine whether or not the Contract transformation should be enacted between two cliques. One simple criterion, applicable if the state vector components all have the same sample space—as they do in the Gaussian case—is to Contract iff

$$\sum_{c[i] \in n(c[j])} \#(s[i, j]) + \sum_{c[i] \in n(c[j'])} \#(s[i, j']) - 2\#(s[j, j'])$$

$$\leqslant \max \left\{ \sum_{c[i] \in n(c[j])} \#(s[i, j]) + \sum_{c[i] \in n(c[j'])} \#(s[i, j']) \right\}.$$

This criterion uses Contract conservatively: it is only employed when, on combining cliques $c[j]$ and $c[j']$, the subsequent use of Thin will make the new clique no bigger.

When all separators are of the same dimension it is employed iff both $c[j]$ and $c[j']$ each have exactly two edges in T.

7. Transformations which approximate

Even when exact techniques are used to simplify a dynamic junction forest, there are times when the forest gradually becomes more and more inefficient. If there is an imperative to keep computational algorithms quick, it then becomes necessary to approximate that forest with ones which process information more quickly.

Obviously such an approximation must be invoked with great care, because an approximation of the distribution of essential states may now look superficially good but with the passage of time may give rise to a distribution of future states which is inadequate. There are two ways to investigate whether this problem is likely to arise. The first, illustrated later in this section, just checks the efficacy of the approximation with typical data streams on which it might be used against an exact calculation. The second way is analytic—we introduce a metric over the distributions on essential states of a given time and use this to judge the approximation. A good metric to use for this purpose is the *Hellinger* metric, given by

$$d_H(p, p') = \int (p^{1/2} - p'^{1/2})^2,$$

where p and p' are, respectively, a density and its approximating density. This is topologically equivalent to the variation metric. An alternative, the *Kullbach–Leibler* separation measure is discussed by Gargoum [5].

The first important point to notice is that all three measures of separation mentioned above are "local". Thus, for example, it is easy to check that if p and p' are densities over variables in a junction tree process J which differs only on a clique c, then $d_H(p,p') = d_H(p_c, p'_c)$, where p_c, p'_c are the marginal densities of c under density p and p', respectively. Also any margin of variables under p or p' will be no further apart than $d_H(p_c, p'_c)$ (see, e.g., Smith [27] for a proof of these results). Thus, in particular, if a margin p_c of $c \in C$ is approximated by p'_c but the process is otherwise left unaltered then we can guarantee that

$$d_H(p_e, p'_e) \leqslant d_H(p_c, p'_c),$$

where p_c and p'_c are, respectively, the true and approximating densities on essential states induced by this substitution. It follows that small approximations of clique margins will give rise to only small changes in distributions over states.

A legitimate concern would be that the repeated use of such approximations might, with time, aggregate errors on essential states. Happily, for problems which are truly stochastic—i.e., for which there is a significant system error in the state equations—distances between true and approximate densities are expected to decrease geometrically with time. So in practice, provided observations are consistent with their predictive distributions, such aggregation tends not to be a problem—see Smith [27] for more details.

Gargoum and Smith [6] and Gargoum [5] discuss several such approximations to be used with Example 3.1. Here we will discuss just one: the most easy to generalise. The *Cut*

transformation of $\{T, C, S, P\}$ to $\{T^*, C^*, S^*, P^*\}$ deletes an edge $s[i, j]$ in the forest T between the two cliques $c[i]$ and $c[j] \in C$ as follows:

$$\text{set } S^* = S \backslash \{s[i, j]\},$$
$$C^* = \left(C \backslash \{c[j]\}\right) \cup \left\{c^*[j]\right\},$$

where $c[j] = (\phi_1[j], \phi_2[j])$, $s[i, j] = \phi_1[j]$, $c^*[j] = \phi_2[j]$ and T^* is defined by setting C^* and S^* defined above as its node and edge set. If $c \in C^*$ $c \neq c^*[j]$ then $p^*(c) = p(c) \in P$. Finally the density of $c^*[j]$ is simply the margin of $\phi_2[j]$ obtained from the density of $c[j]$.

Technically all this transformation does is to replace the density p_j of $c[j]$ by the product $q_j r_j$ where q_j is the density of $\phi_1[j]$ and r_j the density of $\phi_2[j]$. The Hellinger distance of this approximation will be small if $\phi_1[j]$ and $\phi_2[j]$ are almost independent of each other.

In the Gaussian case

$$d_H = \left(1 - \det\left\{1/4\left(2I + K + K^{-1}\right)\right\}\right)^{1/2},$$

where

$$K = \begin{bmatrix} I_1 & A_1 \\ A_2 & I \end{bmatrix}$$

and I are the identity maps of the appropriate dimensions, A_1 is the regression matrix of $\phi_1[j]$ and $\phi_2[j]$ and A_2 the regression matrix of $\phi_2[j]$ and $\phi_1[j]$. So here we can clearly see that d_H is close to zero when A_1 and A_2 are both close to zero—i.e., when learning $\phi_1[j]$ is expected to hardly change the mean vector of $\phi_2[j]$ and vice versa.

Note that, whatever the distribution on essential state, the use of the Cut transformation is expected to be conservative—it prohibits learning about $\phi_1[j]$ from $\phi_2[j]$ and about $\phi_2[j]$ from $\phi_1[j]$—it therefore ignores certain information in its estimation processes but acknowledges that it does this. In particular, it is never expected to inflate its uncertainty about its predictions—something to be avoided in approximate learning systems—see Smith [25].

Fig. 11 gives the junction forests obtained by using the Cut transformation to approximate a forest which was originally a single tree. Forest 1 approximation demands a smaller Hellinger distance than Forest 2 approximation and so has fewer trees. Thus Tree 2 in Forest 1 has split into two degenerate cliques—Tree 5 and Tree 7 in Forest 2—whilst clique $c[5]$ has disappeared completely under the Lop transformation since it is a transmitter clique separated from the essential cliques by the action of Cut. The forests are modeling the spread of nuclear contamination and in this context even when the process has run a long time and has many cliques the size of the trees with appropriate settings of the Hellinger distance, under Cut, they are typically of about this size.

In this simulation we purposely let observed levels of contamination be much larger than forecasts by the model and so there is no mathematical reason why the approximating processes should necessarily remain close to the original one. However, for forecasts of levels of contamination which might prompt action, approximate forecasts never deviated by more than 0.3 of the forecasts associated with the true model—an insignificant deviation as far as determining appropriate action was concerned. So the methodology seems remarkably robust. In particular in many runs at different parameter settings of the model

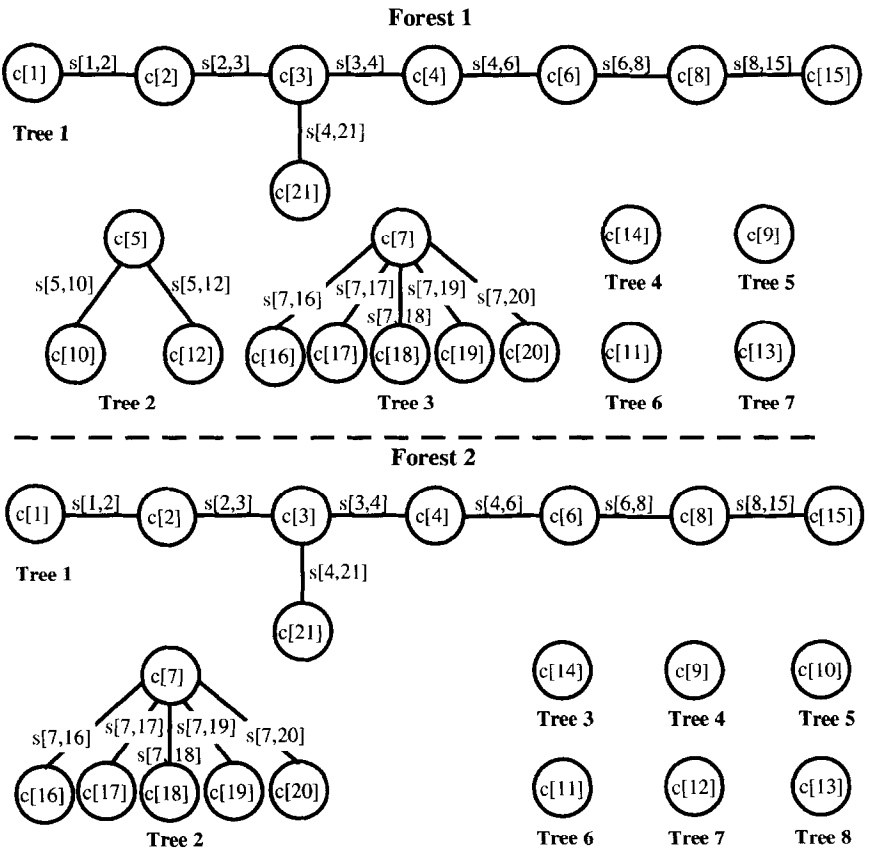

Fig. 11. Two forests where cut is administered at different levels of the Hellinger distance.

approximation error for forecast contamination does not appear to escalate with time even when outlying data is observed.

8. Conclusions

The techniques of dynamic probabilistic learning of the type described in this paper are not just an aid to fast computation. The automated husbandry of belief structures over variables of interest can be fed back to the user to show her how the system is using data to learn about the system. The graphical structures in particular are very useful in this regard. They show how dependencies and independencies are created and destroyed by the passage of time.

The use of structural approximation, like the Cut, is particularly interesting because it mirrors human learning. For in practice we always limit the scope of our statistical model by choosing to disregard data, which might be observed, because we believe that there is little or no chance of it affecting our beliefs significantly and assuming variables

are independent when our evidence to date strongly suggests that this is almost so. Running with an appropriate diagnostic over the prediction space and treating a Bayesian model as an albeit complex null hypothesis—as it is implicit in most current practical Bayesian modeling—is entirely consistent with taking such approximations and, in our belief, underpins any inferential structure which allows for creativity. Certainly as part of a decision support system such dynamic systems are already providing stimuli to creative problem solving in at least one application.

Acknowledgements

We are grateful to our colleagues in the RODOS project for many discussions. This work is funded by the CEC (Contract: F14P-CT95–0007) and EPSRC (Grant no.: GR/K72254). The views expressed in this paper are those of the authors and do not necessarily reflect those of the RODOS project.

Appendix A. Proof of Theorem 5.1

Theorem 5.1. *If the junction tree state* (T, C, S, P) *is valid then so is the junction tree state* (T', C', S', P') *defined in Section* 5.

Proof. Since T is a forest the running intersection property (RIP) [32] ensures that the conditional independence (CI) statements in T can be equivalently coded by a junction graph I_1 which has a compatible ordering of nodes that introduces first the nodes in $B[1]$, then the nodes in $B[2], \ldots$ then the nodes in $B[k]$ and subsequently the remaining nodes in C.

Since by construction T' is also a forest, the RIP also tells us that its ci statements can be equivalently coded by a junction graph I_2 which introduces first the nodes in $C'[1]$, then the nodes in $C'[2], \ldots,$ then the nodes in $C'[k]$ and then the remaining nodes in C' (which are also in C) in the same order as for I_1.

By the d-separation Theorem [15,17,18] since the junction graphs I_1 and I_2 are causal diagrams/influence diagrams and are identical on their shared nodes all the CI statements in I_2 will be valid iff the CI statements implicit in $C'[1], \ldots, C'[k]$ can be deduced from I_1. There are only two such statements. The first is that

$$\coprod_{j=1}^{k} \{c \in C'[j]\}$$

and this is implicit from I_2 since $C'[1], \ldots, C'[k]$ all contain variates in cliques which lie in different trees of the forest in T.

The second class of statements concern the dependencies between variables in the cliques in $C'[j]$, $j = 1, \ldots, n$. But the CI statements in $C'[j]$ are implied by the statement that sets of variables in $B[j]$ are independent of each other given the values of all the variables in cliques adjacent to the sets. This is a direct and trivial consequence of the d-separation Theorem. So our result is proved. □

Appendix B. Proof of Theorem 6.1

Theorem 6.1. *If the junction tree state $J = (T, C, S, P)$ is valid then so is the junction tree $J^* = (T^*, C^*, S^*, P^*)$ defined in Section 6.*

Proof. First check that if T has k disconnected trees $T[1], T[2], T[3], \ldots, T[k]$ and T^* consists of k^* disconnected trees $T^*[1], T^*[2], T^*[3], \ldots, T^*[k^*]$, then $k^* \leqslant k$, $T^*(j^*)$ is a subtree of $T(j)$, $1 \leqslant j^* \leqslant k^*$, $1 \leqslant j \leqslant k$, and no two trees $T^*(j_1^*)$ and $T^*(j_2^*)$ are subtrees of the same tree $T(j)$, $1 \leqslant j_1^*, j_2^* \leqslant k^*$. It is therefore possible to label the nodes in a tree $T(j)$ containing $T^*(j)$ using the running intersection property starting with nodes in the subtree $T^*(j)$ in a way compatible with a directed tree whose undirected version is $T^*(j)$ and each node has exactly one parent (except the root node which has none) $1 \leqslant j \leqslant k$. Continue to label the nodes in tree $T(j)$ so that the labeling in a way compatible with a directed tree whose undirected version is $T(j)$ and each node has exactly one parent (except the root node which has none $1 \leqslant j \leqslant k$).

The junction graph G whose disconnected directed trees have undirected versions $T[1], \ldots, T[k]$ and are compatible with the order above are valid. Since all nodes in the corresponding directed subforest $T^*[1], \ldots, T^*[j^*]$, comprise an ancestor set in G it follows by the d-separation Theorem [17] that G^*—the subjunction graph of G whose undirected version is T^*—is valid. Since C^* contains all the states of the process and the clique margins in C^* agree with those in C, by construction, the result follows. □

References

[1] R.G. Cowell, BAIES—A probabilistic expert system shell with qualitative and quantitative learning, in: J.M. Bernardo, J.O. Berger, A.P. Dawid, A.F.M. Smith (Eds.), Bayesian Statistics 4, Clarendon Press, Oxford, 1992, pp. 595–600.

[2] A.P. Dawid, Applications of a general propagation algorithm for probabilistic expert systems, Statist. Comput. 2 (1992) 25–36.

[3] A.P. Dawid, S.L. Lauritzen, Hyper Markov laws in the statistical analysis of decomposable graphical models, Ann. Statist. 21 (3) (1993) 1272–1317.

[4] S. French, D.C. Ranyard, J.Q. Smith, Uncertainty in RODOS, Research Report 95.10, School of Computer Studies, University of Leeds, UK, 1995.

[5] A.S. Gargoum, Issues in Bayesian forecasting of dispersal after a nuclear accident, Ph.D. Thesis, Statistics Department, University of Warwick, UK, 1998.

[6] A.S. Gargoum, J.Q. Smith, Approximating dynamic Gaussian junction trees, Research Report 279, Statistics Department, University of Warwick, UK, 1994.

[7] P.J. Harrison, C.F. Stevens, Bayesian forecasting (with discussion), J. Roy. Statist. Soc. Ser. B 38 (1976) 205–247.

[8] F. Jensen, F.V. Jensen, S.L. Dittmer, From influence diagrams to junction trees, in: Proceedings 10th Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, 1994, pp. 367–373.

[9] F.V. Jensen, An Introduction to Bayesian Networks, U.C.L. Press, London, 1996.

[10] F.V. Jensen, K.G. Olesen, S.K. Andersen, An algebra of Bayesian belief universes for knowledge-based systems, Networks 20 (1990) 637–659.

[11] U. Kjærulff, A computational scheme for reasoning in dynamic probabilistic networks, in: Proceedings 8th Conference on Uncertainty in Artificial Intelligence, Stanford, CA, 1992, pp. 121–129.

[12] S.L. Lauritzen, Propagation of probabilities, means and variances in mixed graphical association models, J. Amer. Statist. Assoc. 87 (1992) 1098–1108.

[13] S.L. Lauritzen, Graphical Models, Oxford University Press, Oxford, 1996.
[14] S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems (with discussion), J. Roy. Statist. Soc. Ser. B 50 (1988) 157–224.
[15] S.L. Lauritzen, A.P. Dawid, B.N. Larsen, H.G. Leimer, Independence properties of directed Markov fields, Networks 20 (1990) 491–505.
[16] K.V. Mardia, J.T. Kent, J.M. Bibby, Multivariate Analysis, Academic Press, London, 1979.
[17] J. Pearl, Probabilistic Inference in Intelligent Systems, Morgan Kaufmann, San Mateo, CA, 1988.
[18] J. Pearl, T.S. Verma, The logic of representing dependencies by directed graphs, in: Proceedings 6th National Conference on Artificial Intelligence (AAAI-87), Seattle, WA, 1987, pp. 374–379.
[19] C.M. Queen, Using a multi-regression dynamic model to forecast sales in a competitive market, The Statistician 43 (1) (1994) 87–98.
[20] C.M. Queen, Model elicitation in competitive markets, in: S. French, J.Q. Smith (Eds.), The Practice of Bayesian Analysis, Arnold, London, 1997, pp. 229–243.
[21] C.M. Queen, J.Q. Smith, Multi-regression dynamic models, J. Roy. Statist. Soc. Ser. B 55 (4) (1993) 849–870.
[22] C.M. Queen, J.Q. Smith, D.M. James, Bayesian forecasts in markets with overlapping structures, Internat. J. Forecasting 10 (1994) 209–233.
[23] R.D. Shachter, Evaluating influence diagrams, in: A.P. Bau (Ed.), Reliability and Quality Control, Elsevier, North-Holland, Amsterdam, 1986, pp. 321–344.
[24] J.Q. Smith, Influence diagrams for statistical modelling, Ann. Statist. 17 (1989) 654–672.
[25] J.Q. Smith, Discussion of "Learning in probabilistic systems" by J. Spiegelhalter and R.G. Cowell, in: J.M. Bernardo, J.O. Berger, A.P. Dawid, A.F.M. Smith (Eds.), Bayesian Statistics 4, Clarendon Press, Oxford, 1992, pp. 460–463.
[26] J.Q. Smith, Handling multiple sources of variation using influence diagrams, European J. Oper. Res. 86 (1995) 189–200.
[27] J.Q. Smith, Bayesian Approximations and the Hellinger metric, J. Roy. Statist. Soc. Ser. B, to appear.
[28] J.Q. Smith, S. French, Bayesian updating of atmospheric dispersion models for use after an accidental release of radioactivity, The Statistician 42 (5) (1993) 501–512.
[29] J.Q. Smith, S. French, D.C. Ranyard, An efficient graphical algorithm for updating estimates of the dispersal of gaseous waste after an accidental release, in: A. Gammerman (Ed.), Probabilistic Reasoning and Bayesian Belief Networks, Alfred Walker, 1995, pp. 125–140.
[30] D.J. Spiegelhalter, S.L. Lauritzen, Sequential updating of conditional probabilities on directed graphical structures, Networks 20 (1990) 579–605.
[31] D.J. Spiegelhalter, A.P. Dawid, S.L. Lauritzen, R.G. Cowell, Bayesian analysis in expert systems, Statistical Science 8 (1993) 219–246.
[32] R.E. Tarjan, M. Yannakakis, Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs, SIAM J. Comput. 13 (1984) 566–579.
[33] A. Thomas, D.J. Spiegelhalter, W.R. Gilks, BUGS: A program to perform Bayesian inference using Gibbs sampling, in: J.M. Bernardo, J.O. Berger, A.P. Dawid, A.F.M. Smith (Eds.), Bayesian Statistics 4, Clarendon Press, Oxford, 1992, pp. 837–842.
[34] M. West, P.J. Harrison, Bayesian Forecasting and Dynamic Models, Springer, New York, 1996.