Hierarchical Junction Trees: Conditional Independence Preservation and Forecasting in Dynamic Bayesian Networks with Heterogeneous Evolution

Roberto O. Puch[1], Jim Q. Smith[2], and Concha Bielza[3]

[1] School of Crystallography, Birkbeck College, University of London, London WC1E 7HX, UK
[2] Department of Statistics, University of Warwick, Coventry CV4 7AL, UK
[3] Artificial Intelligence Department, Technical University of Madrid, Madrid 28660, Spain

Abstract. Propagation in decomposable Bayesian networks with junction trees is inferentially efficient: no conditional independence in the Bayesian network is ignored in the junction tree construction and in any propagation task. For non-decomposable Bayesian networks, the junction tree construction uses moralisation and triangulation that ignore some of the conditional independence. The junction tree, therefore, trades inferential efficiency with generality: it can be used to compute the distribution of any set of target nodes given any set of conditioning nodes.

In this chapter inferential efficiency for non-decomposable Bayesian networks is addressed. We present the hierarchical junction tree, a framework that transparently represents the conditional independence in the Bayesian network. We discuss propagation tasks where conditional independence is not ignored in the construction of the hierarchical junction tree and in the propagation tasks. We also discuss their use for efficient exact forecasting in dynamic Bayesian network with heterogeneous evolution.

1 Introduction

Fast probability propagation in a decomposable Bayesian network (BN) is often performed through a secondary structure called the junction tree (JT) [6,3,2]. Conditional independence statements encoded in the BN are used for constructing this structure. A JT provides a general facility for computing probabilities of any set of variables of interest, subsequently called target variables, given the values of any other set of observed variables, subsequently called conditioning variables. However for non-decomposable BNs the JT framework can only provide this facility at the cost of ignoring some of the conditional independence encoded in that BN. Conditional independence statements are ignored both in the moralisation step, where parents of the same child are joined with an undirected edge and then edge direction

J. A. Gámez et al. (eds.), *Advances in Bayesian Networks*

is dropped, and also in the triangulation step, where undirected edges are added. The addition of these undirected edges, the so-called fill-in's, has the tendency to create large cliques in the JT. The JT propagation algorithm can then become computationally inefficient. This inefficiency can become critical particularly in dynamic settings.

To alleviate these problems, specifically in a non-dynamic setting, Madsen & Jensen [13,14] proposed the Lazy propagation (LP) method. This method records extra conditional independence which had been coded in the BN but not retained in the JT through the potentials of each clique. They preserve each edge direction by recording the potentials as conditional probability distributions, and clique potentials are kept factorised. The LP method uses therefore a hybrid representation, both graphical and algebraic, to store conditional independence statements required for efficient propagation.

It is now timely to stand back and view propagation in a new light. In this article we present a formal framework, the hierarchical junction tree (HJT), for probability propagation that transparently and consistently encodes the edge direction and potential factorisation in the BN. The HJT consistency of graph and potentials representation allows us to prove separation theorems which implies that the HJT is constructed without loss of conditional independence.

For propagation tasks where the conditioning variables are ancestors of the target variables and the ancestral graph of the conditioning variables is decomposable, all conditional independence statements encoded in the BN are transferred to the HJT. Examples of these propagation tasks arise in dynamic models where the primary interest is in forecasting, and so the level of generality that the JT provides is often not required. This setting often occur in the development of decision support systems for emergency planning under uncertainty [16,17]. In decision support systems in the case of a nuclear accident, for example, counter-measures cannot affect the past contamination levels but only future received radiation doses. In addition the information that is usually received is on contamination levels in the past. Although forecasting is a less complex computational task, in this type of problems forecasting is required on-line and as soon as possible so that counter-measures are implemented. In contrast, smoothing is done off-line.

In most dynamic settings, it is natural to construct the BN so that variables appearing earlier in the BN tend to be learned before the variables appearing later in the network. In this setting we also have new descendant variables added with time, and we forecast the future in the light of observing the past. Thus it is typical for the target variables to be descendants of the conditioning variables. Furthermore the use of JT algorithms in settings where new descendant nodes are continuously added to the net has some inherent problems. First, the standard JT algorithms, that is the JT with the Hugin architecture [1], provide no facility to move variables out of the system as they are learned because this is not required for static networks.

So, the number of variables in the system and cliques in the tree steadily increase. Lazy propagation and Nested JTs inherit this feature because they operate on the standard JT. Second, as new variables arrive with time, on-line triangulation is required. This tends to progressively increase the average clique size and further undermines efficiency. Although LP is not highly dependant on the triangulation, on-line triangulation is still required because LP operates on a standard JT.

In this article we also propose the HJT as a efficient framework for exact forecasting in dynamic BNs with heterogeneous evolution. These issues were first addressed in articles for time series with homogeneous time-slices by Kjærulff [10] and Kanazawa *et al.* [7]. However, in the methodology described below we do not assume time homogeneity of structure. The HJT is constructed from the BN without loss of conditional independence when the ancestral graph of the conditioning variables is decomposable [15]. Only this ancestral graph needs to be triangulated when it is not decomposable. In contrast, the whole graph needs to be triangulated in the JT framework. In addition the HJT decreases its size as variables are learned to offset any increase in the descendants, see Section 5.4.

Having developed this framework we can start to formally address and compare issues of efficiency of competing propagation algorithms. Perhaps more importantly we can tailor these algorithms to the predictive needs of the system, often necessary in propagation in dynamic BNs.

In Section 2 BNs are reviewed and the new concept of ancestrality partition is introduced. We review JTs in Section 3 while in Section 4 the using JTs for forecasting in dynamics BNs with heterogeneous evolution is presented. We introduce HJTs in Section 5 and their construction is illustrated through an example. Also in this section we introduced the h-separation criterion for HJTs and a method for propagating probabilities in HJTs. In the last section conclusions are given and future research challenges are discussed.

2 Bayesian Networks

We recall that a *Bayesian network* is a pair (G, P) consisting of a directed acyclic graph $G = (V, E)$ and a probability distribution P. Set V is the index of a set of variables $\{X_v\}_{v \in V}$. Probability distribution P factorises according to G, that is, $P(x_V) = \prod_{v \in V} Pr(X_v = x_v | X_{Pa(v)} = x_{Pa(v)})$ where $Pa(v)$ is the set of parents of node v in the node set V of G.

In the transition from the BN to the HJT, we divide the BN into layers. This division relies on the following concept. A *collider* in a BN is a node having at least two parents which are not joined by an edge. The layers of the BN is then based on the configuration of the colliders. A decomposable BN can be defined as a BN that has no colliders. Some of the nodes of the HJT will be *families* of the nodes of a BN, that is, sets of the form $Fa(v) = \{v\} \cup Pa(v)$.

In fact they will be *maximal* families where family $Fa(v)$ is maximal if it is not contained in another family.

The first step in the transition from the BN to the HJT is the identification of the ancestrality partition of the BN. In the following we will introduce and discuss this new concept.

Let $B = (G, P)$ be a BN where $G = (V, E)$. The *ancestrality partition of* B is the partition of V,

$$\{I(1), H(1), I(2), \ldots, I(m-1), H(m-1), I(m)\}.$$

The elements of this partition are defined as,

$I(1)$ $=\{v : v$ is not a collider and has no collider ancestors$\}$

$H(1)=\{v : v$ is a collider and has no collider ancestors$\}$,

$I(i)$ $=\{v : v$ is not a collider, has at least one collider ancestor in $H(i-1)$ and has no collider ancestors in $V \setminus [H(1), \ldots, H(i-1)]\}$.

$H(i)$ $=\{v : v$ is a collider, has at least one collider ancestor in $H(i-1)$ and has no collider ancestors in $V \setminus [H(1), \ldots, H(i-1)]\}$.

$I(i)$ is defined if $H(i-1)$ exists. $H(i)$ is defined if $H(i-1)$ exists and if it is not empty.

The ancestrality partition organises the BN into layers determined by the pattern of colliders. The first layer consists of $I(1)$, the second layer consists of $H(1) \cup I(2)$. In general, the i-th layer consists of $H(i-1) \cup I(i)$, $i = 2, \ldots, m$. These layers help to identify decomposable sections of the original BN from which the JTs of the HJT are constructed, see Section 5.

We can construct the ancestrality partition of a BN given a total order consistent with the partial order induced by the BN, by organising the nodes in layers. Let $(v(1), v(2), \ldots, v(n))$ be a total order and $L(v(i))$ denote the layer where $v(i)$ is. Starting with $v(1)$ the sequential rule is:

1) If $Pa(v(i)) = \emptyset$, $L(v(i)) = 1$
2) If $Pa(v(i)) \neq \emptyset$ and $v(i)$ is not a collider,
   $L(v(i)) = max\{L(u) : u \in Pa(v(i))\}$
3) If $Pa(v(i)) \neq \emptyset$ and $v(i)$ is a collider,
   $L(v(i)) = max\{L(u) : u \in Pa(v(i))\} + 1$.

Then the elements of $I(1)$ are those nodes v with $L(v) = 1$. The elements of $H(i-1)$ are those colliders v with $L(v) = i$ and the elements of $I(i)$ are those non-colliders v with $L(v) = i$.

The ancestrality partition of the BN given in Figure 1 is,
$I(1) = \{a, b, c, d, e, f, g, h, j, l\}$, $H(1) = \{i\}$, $I(2) = \{k\}$, $H(2) = \{m\}$, $I(3) = \emptyset$.

Note that the ancestrality partition is invariant under Markov equivalent classes. Two BNs in the same equivalent class have the same collider structure and therefore the same ancestrality partition.

In the dynamic setting introduced in Section 4, new descendants are added sequentially to a BN as time passes. This sequential addition gives a total order of the nodes that is consistent with the partial order induced by the

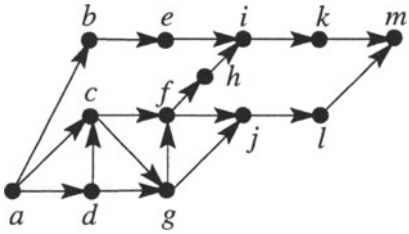


Fig. 1. BN B_2 for the ancestrality partition example

BN, and therefore this order can be used for constructing the ancestrality partition.

3 Junction Trees

In this section we review standard JTs, that is, JTs with the HUGIN architecture [1]. A junction tree for a vector for a potential ϕ_V with *universe* V, is a pair $T = (\mathsf{T}, \phi_V)$. T is a tree of subsets of V such that the so-called JT property holds, that is, the intersection of two sets C_1 and C_2 is contained in all the sets in the unique path between C_1 and C_2. The intersection $C_1 \bigcap C_2$ is called the separator of edge $\{C_1, C_2\}$. ϕ_V is a potential on X_V, where a potential ϕ_U is a function whose domain is the state space of the variable X_U and whose counter domain is the non-negative real numbers. Potential ϕ_V factorises according to T such that,

$$\phi_V = \frac{\prod_{C \in \mathcal{C}} \phi_C}{\prod_{S \in \mathcal{S}} \phi_S},$$

where $\mathcal{C}$ is the set of nodes of T and $\mathcal{S}$ the set of separators. A directed JT is a JT with directed edges.

We can construct a JT $T = (\mathsf{T}, \phi_V)$ from a given decomposable BN $B = (G, P)$. The nodes of the tree T are cliques of the undirected version of G. The potentials ϕ_C are formed from the product of conditional probabilities $Pr(X_v = x_v | X_{Pa(v)} = x_{Pa(v)})$, seen as a function of $(x_v, x_{Pa(v)})$, where $\{v\} \cup Pa(v) \subseteq C$. The potentials ϕ_S for the separators are initialised to 1. If the BN is not decomposable but its moral graph is chordal we can still construct a JT from the moral graph, but some conditional independence is lost in the moralisation step. If the BN is not decomposable and its moral graph is not chordal, then its moral graph is made chordal through a triangulation step before we construct the JT and this step can be computationally expensive. For a detailed study on triangulation see [8,9] or [12]. Triangulating the moral graph requires us to ignore some of the conditional independence encoded in the BN because we need to add extra edges, the so-called fill-in's. In this case a product of conditional distributions may be assigned to one clique, and therefore losing part of the factorisation encoded in the BN.

Propagation in JTs within the Hugin architecture is based on the operation of a sum-flow from a node to one of its neighbouring nodes. Let C_i and C_j be neighbours with separator $S = C_i \bigcap C_j$, and let ϕ_{C_i}, ϕ_{C_j} and ϕ_S be their corresponding current potentials. A sum-flow is performed as follows,
1) compute the potential $\phi'_S = \sum_{C_i \setminus S} \phi_{C_i}$;
2) assign $\phi'_{C_j} = \phi_{C_j} * \frac{\phi'_S}{\phi_S}$;
3) assign ϕ'_S to S.

Propagation in the Hugin architecture is done in two steps. Let us suppose that we learn $X_a = x'_a$, or for short, that we learned X_a. First, we choose a clique that contains a, then we multiply the potential of this clique by the finding f_{a,x'_a} for x'_a, where a finding for x'_a is a potential with domain $Sp(X_a)$ that maps x'_a to 1 and 0 for other elements of $Sp(X_a)$. Second, we schedule sum-flows by choosing a clique as a root and then scheduling sum-flows first from the leaves of the tree to the root, the so-called Collect Evidence schedule, and then scheduling sum-flows from the root to the leaves, the so-called Distribute Evidence schedule. After these two schedules have been performed, the JT is sum-consistent, that is, the potentials on the nodes and the separators hold the marginal potentials of the potential P of the JT. For a description of the Hugin architecture see [5,2], and for a more technical description see [3].

4 Forecasting in the Dynamic Setting Using Junction Trees

In this section we describe how JTs can be used for forecasting in a dynamic setting. We first address how a JT can be constructed sequentially and then demonstrate how it can be used for forecasting. The sequential construction is demonstrated for $t = 1$ and the transition from $t = 1$ to $t = 2$. However, the second transition is trivially identical to constructing a JT at time $t+1$ from a JT valid at time t. Figure 2 displays the BNs B_1 for time $t = 1$ and B_2 for time 2. BN B_1 is surrounded by the inner dashed lines and B_2 is surrounded by the outer dashed lines. Descendant nodes k, l and m were sequentially added into B_1 to obtain B_2.

The construction of the JT T_1 from BN B_1 is done through a standard procedure: the BN is moralised, the moral graph is then triangulated and finally a tree of cliques is constructed from the triangulated graph. In the construction of JT T_1, displayed in Figure 3, the fill-in's are: $\{a, f\}$, $\{a, g\}$, $\{b, f\}$ and $\{e, f\}$.

At time $t = 2$, the nodes k, l and m are added to B_2. If we wish to compute the marginals for X_m on B_2, we need to construct a JT for B_2. One choice would be to start over again and construct a JT for B_2. Another choice would be to use the fill-in's in the triangulation of the moral graph of B_1 as a starting point for the triangulation of the moral graph of B_2. Notice that in both choices on-line triangulation is required. The second choice is

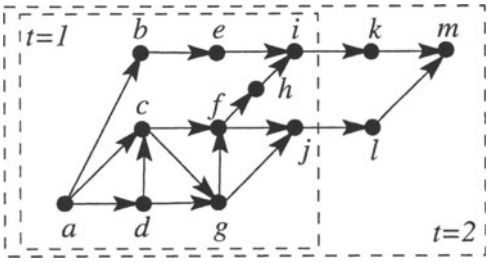


Fig. 2. The BN B_1 at time $t = 1$ is surrounded by the inner dashed box; the BN B_2 at time $t = 2$ is surrounded by the outer dashed box

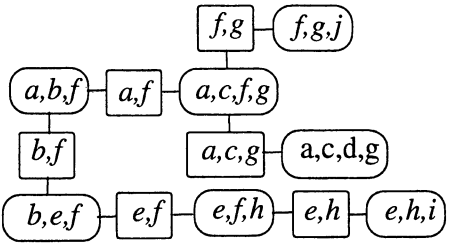


Fig. 3. The JT T_1 constructed from BN B_1 in Figure 2

more computationally efficient because part of the moral graph of B_2 has been already triangulated with the triangulation of the moral graph of B_1. The JT T_2, constructed in this fashion, is displayed in Figure 4. The extra fill-in's are $\{b, g\}$, $\{e, g\}$, $\{f, i\}$, $\{g, i\}$, $\{i, j\}$ and $\{i, l\}$.

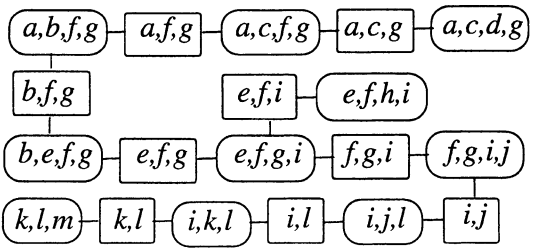


Fig. 4. The JT T_2 constructed from BN B_2 in Figure 2

In addition to the triangulation process being computationally expensive, the sequential on-line triangulation results in a sequential increment of the node/clique sizes. In T_2, for example, node $\{a, b, f\}$ of T_1 is of size 3 and it is included in node $\{a, b, f, g\}$ of T_2 which is of size 4. Note as well that nodes $\{b, e, f\}$ and $\{e, f, g\}$ in T_1 are joined together into node $\{a, e, f, g\}$ in T_2.

Once we have constructed a JT, we can use it for forecasting. The JT presents an inefficiency in that the fill-in's added in the triangulation process

may be superfluous once we learn a variable. In the JT T_2, for example, if we learn that $X_a = x'_a$ and we remove a from B_2, then the fill-in's added because of the cycle (a, c, f, h, e, b) in the moral graph of B_2 are no longer needed. A solution would be to construct a JT for the reduced BN, but this process is computationally expensive. We will see in Section 5.4, that the HJT does not have this problem because no fill-in's are added at all.

5 Hierarchical Junction Trees

5.1 Definition

A *hierarchical junction tree* (HJT) $\mathcal{H}$ for a potential ϕ_V with universe V is a directed rooted tree $\mathcal{H} = (T, \mathcal{F})$ where the edge direction is towards the root. The root is a directed JT for ϕ_V. The node set is $T = \{T_1, T_2, \ldots, T_p\}$ where each node T_i, $1 \leq i \leq p$, is a directed JT. The ordered set (T_i, T_j) is in the edge set $\mathcal{F}$ if T_i is a directed JT for ϕ_C where C is a node in the directed JT T_j. The node C is then called a covering node in T_j and a cover of T_i.

The covering nodes of a JT T' in the HJT act as dummy nodes. In fact, the potential associated with a covering node has a JT representation given by the parent T of T' in the HJT. The children of covering nodes are called *upper doors* and the separator of a covering node with an upper door is called an *elevator*. We say that a HJT is sum-consistent if all of its JTs are sum-consistent.

It is important to note that [11] first introduced the idea of JTs with nodes having JTs representations. He proposed the Nested JTs method to alleviate the loss of conditional independence in the construction process of the standard JT, where by standard we mean the JT with the Hugin architecture [1] without the Nested JTs and Lazy propagation methods. The motivation of the HJT is different as it is intended as a propagation framework with preservation and transparent representation of conditional independence.

5.2 Construction

In this section we describe the construction of a HJT from a BN for both the static and dynamic cases. In the first part the construction of the static case is described, which is also a description of the construction of a HJT for the first time step in the dynamic case. In the second part the we show the HJT update given that new nodes have entered the systems. Although the description is from time step 1 to time step 2, the update from time t to time $t + 1$ is analogous. This construction is illustrated with the BNs B_1 and B_2 in Figure 2.

Construction of $\mathcal{H}_1$

In this part we present the general steps for constructing a HJT $\mathcal{H}$ for a BN B. We also illustrate this steps to construct HJT $\mathcal{H}_1$ from BN B_1.

Step 1: Ancestrality partition identification. In general the ancestrality partition is of the form

$$\{I(1), H(1), I(2), \ldots, I(m-1), H(m-1), I(m)\}.$$

We identify this partition using the ancestrality-partition algorithm introduced in Section 2.

In our example, a total order consistent with the partial order induced by BN B_1 is $\{a, b, c, d, e, f, g, h, j, i\}$. The ancestrality partition for B_1 is then: $I_1(1) = \{a, b, c, d, e, f, g, h, j\}$, $H_1(1) = \{i\}$, $I_1(2) = \emptyset$.

Step 2: The construction of the directed JT for Layer 1. This JT is constructed from the BN $B_{I(1)}$ induced in the original BN B by the set $I(1)$ in the ancestrality partition. The nodes in this JT are the maximal families in $B_{I(1)}$. The edges are constructed as follows. A maximal family $Fa(v_1)$ is a parent of maximal family $Fa(v_2)$ if w_1 is a parent of w_2 in $B_{I(1)}$ where $Fa(w_1) \subseteq Fa(v_1)$ and $Fa(w_2) \subseteq Fa(v_2)$, but in most cases v_1 is a parent of v_2. Note that if $B_{I(1)}$ is not a connected BN, a junction forest will be constructed using this algorithm. Each conditional probability distribution $P(X_v|X_{Pa(v)})$ is assigned to the node that contains $Fa(v)$, and separator potentials are initialised to 1.

Let us denote by T_1^1 the JT constructed from the connected BN $B_{I_1(1)}$ displayed in Figure 5. JT T_1^1 is shown in Figure 6. For example, set $\{a, d, c\}$ is maximal family $Fa(c)$ and so, it is a node of T_1^1. Set $Fa(g) = \{d, c, g\}$ is also a node of T_1^1, but $Fa(d) = \{a, d\}$ is not a node because it is a subset of $Fa(c)$. The edge set of T_1^1 is constructed from the edge set of $B_{I_1(1)}$. For example, $Fa(c)$ is a parent of $Fa(g)$.

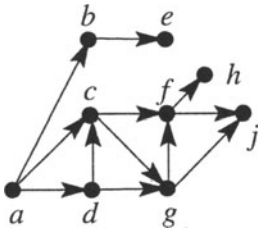


Fig. 5. BN $B_{I_1(1)}$ induced by the set $I_1(1)$ on the BN B_1 in Figure 2

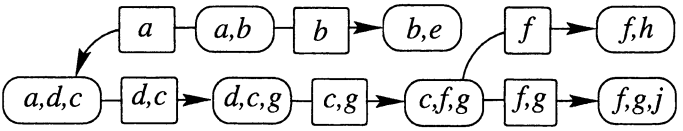


Fig. 6. The directed JT T_1^1 constructed from $B_{I_1(1)}$

Step 3: Construction of the JTs for Layer 2. In general, the JTs in Layer n, $n > 2$ are constructed in three stages. In the first stage JTs are constructed for the BN induced by $Pa(H(n-1) \cup H(n-1) \cup I(n)$. In the second stage covering nodes are identified as the universes of the JTs in Layer $n-1$, and in the third stage covering nodes are connected with directed edges to the JTs constructed in stage 1.

In our example, $Pa(H_1(1) \cup H_1(1) \cup I_1(2) = \{e, h, i\}$ and so, there is one JT consisting of a single node. There is also one covering, $\{a, b, c, d, e, f, g, h, j\}$ which is the universe of the only JT T_1^1 in Layer 1. We joined these two nodes with a directed edge from the covering node to the upper door because their intersection is not empty. we also assign potential $P(X_i|X_{e,h})$ to upper door $\{e, h, i\}$ and initialise the potential of its separator $\{e, h\}$ to 1. The resulting JT is displayed in Figure 7. Notice that covering node $\{a, b, c, d, e, f, g, h, j\}$ is not given a potential because its potential is given by JT T_1^1.

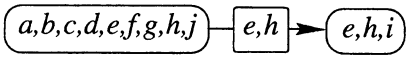


Fig. 7. The directed JT T_1^2 in Layer 2

Step 4: Construction of $\mathcal{H}_1$ edge set. We recall that the nodes of a HJT are JTs. In Steps 2 and 3 the nodes of the HJT were constructed in this step we construct the edge set. The general rule is to join JT T with a directed edge toward T' if T' has a node which is the covering node of T. In our example $\mathcal{H}_1$ has two nodes T_1^1 and T_1^2, and the latter JT contains the covering node of T_1^1. The edge set of $\mathcal{H}_1$ consists of edge (T_1^1, T_1^2). Figure 8 displays HJT $\mathcal{H}_1$.

Construction of $\mathcal{H}_2$

Nodes k, l and m are sequentially added to B_1 to obtain B_2, see Figure 2. In the following steps we update HJT $\mathcal{H}_1$ by accounting for the addition of k, l and m.

Step 1: Ancestrality partition update. The ancestrality partition is sequentially constructed using the algorithm introduced in Section 2 which

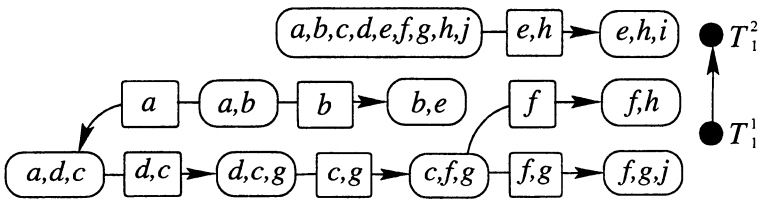


Fig. 8. Hierarchical junction tree $\mathcal{H}_1$

uses a total order of the nodes of the BN. Since the new nodes are descendants of the nodes already in the BN, they can be added at the end of the totally ordered sequence of nodes and apply the ancestrality partition algorithm.

In our example, node k is not a collider and its only parent is $i \in H(1)$, therefore $I_2(2) = \{k\}$. Node l is also not a collider and its only parent is $j \in I_1(1)$ and so, $I_2(1) = \{a, b, c, d, e, g, f, h, j, l\}$. Node m is a collider with parents $k \in I_2(2)$ and $l \in I_2(1)$, thus $H_2(2) = \{m\}$. The updated ancestrality partition is,

$I_2(1) = \{a, b, c, d, e, g, f, h, j, l\}$, $H_2(1) = \{i\}$, $I_2(2) = \{k\}$, $H_2(2) = \{m\}$.

Step 2: HJT update. We recall that non-covering nodes in the HJT are maximal families of the BN and that edges are also inherited from this BN. The arrival of a node is therefore translated to adding a new non-covering node in the HJT. If a new node is not a collider only a new non-covering node is added to an existing JT, while if a new node is a collider, a new JT has to be created to accommodate the extra layer added by this collider.

In our example, the addition of node k to B_1 add the non-covering node $Fa(k) = \{i, k\}$ to T_1^2 to obtain T_2^2. Since k is in Layer 2 we add $Fa(k)$ to T_1^2 with an edge from $Fa(i)$ to $Fa(k)$ because i is the parent of k. The addition of l introduces the non-covering node $Fa(l) = \{j, l\}$ to T_1^1 to T_2^1 because l is in $I_2(1)$.

Adding node m requires creating new JT T_2^3 because m is a collider. The upper door for this node is $Fa(m) = \{k, l, m\}$ and intersects JT T_2^2, thus $\{a, b, c, d, e, f, g, h, i, j, k, l\}$ is a covering node in T_2^3. In this JT we also add directed edge from this covering node to upper door $Fa(m)$. Potential $P(X_m|X_k, X_l)$ is assigned to $Fa(m)$ and separator $\{k, l\}$ is initialised to 1.

We have introduced new node T_2^3 and therefore the edge set has to be updated. The new HJT has new directed edge $\{T_2^2, T_2^3\}$. Updated $\mathcal{H}_2$ is displayed in Figure 9.

5.3 Separation

The construction of the HJT preserves the edge direction of the originating BN through the edge direction of the directed JTs and the edge direction

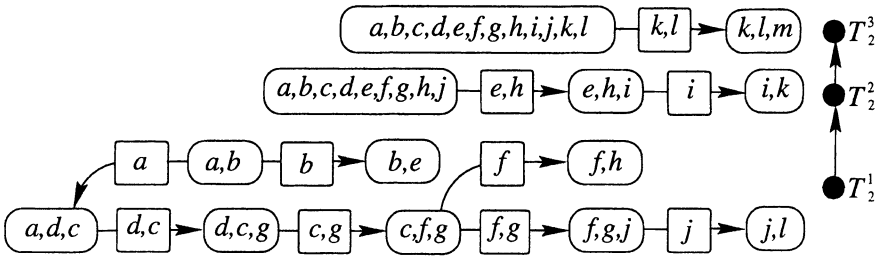


Fig. 9. Hierarchical junction tree $\mathcal{H}_2$

of the HJT joining the JTs. This provides a basis for translating Pearl's d-separation criterion [18] that enables to determine the conditional independence statements implied by the BN to to a criterion that we call *h-separation criterion* that facilitate reading the conditional independence statements implied by the HJT, [15]. In this section we describe this new criterion and and illustrate it using HJT $\mathcal{H}_2$ in Figure 9 constructed from BN B_2 in Figure 1. For a detailed discussion of the h-separation criterion see [15].

The edge direction in the originating BN is mostly preserved through the edge direction of the JTs. For example, edge $(\{a,b\},\{b,e\})$ in $\mathcal{H}_2$ corresponds to (b,e) in B_2. However, edges from a collider's parents to that collider are not directly visible in the HJT. We called these edges *meta edges*. More specifically, if w is a collider and v is one of its parents, the corresponding meta edge to (v,w) is $(Fa(v),Fa(w))$. For example meta edge $(\{b,e\},\{e,h,i\})$ corresponds to (e,i). Note that $\{b,e\}$ is in JT T_2^1 while $\{e,h,i\}$ is in T_2^2.

Each edge in a BN corresponds to either an edge in a JT or a meta edge, except for non-maximal families. For a proof see [15] page 121. For example, edge (b,e) corresponds to edge $(\{a,b\},\{b,e\})$ and edge (e,i) corresponds to $(\{b,e\},\{e,h,i\})$. $Fa(a)$ is not a maximal family and so, edges (a,b), (a,c) and (a,d) correspond to $(\{a,b\},\{a,d,c\})$. This exception has no effect on the bijection between the set of conditional independence statements encoded in a BN and the set of conditional independence statements encoded in its corresponding HJT.

The introduction of meta edges gives a natural extension to parents and children of nodes. More formally, for either an edge in a JT of a HJT or a meta edge (C_1,C_2), C_1 is a *meta parent* of C_2 and C_2 is a *meta child* of C_1. For example, $\{b,e\}$ is a meta parent of $\{e,h,i\}$.

Undirected paths in the originating BN can now be mapped to undirected paths in the HJT. We called these paths *undirected meta paths* because meta edges are allowed in them. For example, path $(b,e,i,h,f,c\}$ is mapped to meta path $(\{a,b\},\{b,e\},\{e,h,i\},\{f,h\},\{c,f,g\},\{d,c,g\},\{a,d,c\})$. Note that this path contains meta edges $(\{b,e\},\{e,h,i\})$ and $(\{f,h\},\{e,h,i\})$. Analogously we can define *directed meta paths* by allowing directed meta edges in directed paths. Path $(\{e,h,i\},\{i,k\},\{k,l,m\})$ is a directed meta path.

Meta descendants can now be defined using directed meta paths. A non-covering node C_2 is a *meta descendant* of a node C_1 if there is a directed meta path from C_1 to C_2. Node $\{k,l,m\}$ is a meta descendant of $\{e,h,i\}$.

We need undirected paths that connect elements of the universe of a HJT to read conditional independence from that HJT. More specifically, for two elements v and w of the universe V of a HJT, a $\{v,w\}$ meta path is a meta path from a node that contains v to a node that contains w. Meta path $(\{a,b\}, \{b,e\}, \{e,h,i\}, \{f,h\}, \{c,f,g\}, \{d,c,g\}, \{a,d,c\})$ is actually a $\{b,c\}$ meta path.

In Pearl's d-separation theorem head-to-head nodes in an undirected path play an important role. Undirected meta paths also have head-to-head nodes. In the meta path above, node $\{e,h,i\}$ is a head-to-head because of meta edges $(\{b,e\},\{e,h,i\})$ and $(\{f,h\},\{e,h,i\})$ are directed towards this node.

In the construction of a HJT in Secion 5.2 we saw that non-covering nodes are families $Fa(v) = \{v\} \cup Pa(v)$ in the originating BN. If we do not have the BN, we can still write a non-covering node C in this form. In fact, $Pa(v) = \cup_r S_r$ where S_r are separators of C shared with its meta parents in the HJT, and $\{v\} = C \setminus Pa(v)$. For example node $\{b,e\}$ in T_2^1 can be written as $\{e\} \cup Pa(e)$ where $Pa(e) = \{b\}$, because $\{b\}$ is the only separator with its parent $\{a,b\}$. For a no-covering node $C = Fa(v)$, v is called the *head* of C and it is denoted by $\widehat{C}$. For example, the head of $\{b,e\}$ is e. Note that writing non-covering nodes as families gives a basis for reconstructing a BN from a HJT. In fact, there is a bijection between the set of maximal families in a BN and the set of non-covering vertices in its corresponding HJT. For a proof see [15] page 113.

In the d-separation criterion the concept of blocking undirected paths between two nodes is used. Analogously, in the h-separation criterion uses blocking of undirected paths. Formally, a $\{v,w\}$ meta path is *blocked* by a set Z, $v,w \notin Z$, if either

(a) there is a separator in this path contained in Z, or

(b) there is a head-to-head C in this path such that $\widehat{C} \cup \left(\cup_j \widehat{C}_j\right)$ does not intersect Z, where C_j is a meta descendant of C.

For example, path $(\{a,b\},\{a,d,c\})$ is blocked by $\{a\}$. Meta path $(\{a,b\}, \{b,e\}, \{e,h,i\}, \{f,h\}, \{c,f,g\}, \{d,c,g\}, \{a,d,c\})$ is also blocked by $\{a\}$ because $\{i,k,m\}$ does not intersect $\{a\}$. Set $\{i,k,m\}$ is the set of heads of $\{e,h,i\}$ $\{i,k\}$ and $\{k,l,m\}$, and sets $\{i,k\}$ and $\{k,l,m\}$ are the meta descendants of $\{e,h,i\}$.

The definition of h-separation is based on the concept of blocking paths. For subsets A, B and C of the universe V of a HJT, A and B are *h-separated* by Z if each $\{a,b\}$ meta path, $a \in A$ and $b \in B$ are h-separated by Z. We can now state the h-separation theorem.

H-separation criterion. $X_A \perp\!\!\!\perp X_B | X_C$ if A and B are h-separated by C.

Using this criterion we deduce that $X_a \perp\!\!\!\perp X_c | X_b$ in HJT $\mathcal{H}_2$, because all $\{a, b\}$ paths are blocked by $\{a\}$.

Theorem. The set of conditional independence statements read in a BN using the d-separation criterion is the same as the set of conditional independence statements read in its corresponding HJT using the h-separation criterion. **Proof:** see [15] page 147.

The translation of the d-separation gives a proof that the HJT can be constructed without loss of conditional independence [15]. In particular note that no triangulation was required to construct the HJT.

The Madsen & Jensen's Lazy propagation method [13] also preserves the conditional independence from the originating BN in the construction of their JT. They keep the directionality of the potentials by recording them as conditional distributions, they keep the potentials for the cliques factorised, and they set the separator potentials to 1. The factorisation of the joint distribution in the BN and the JT coincide and so does the conditional independence encoded in these graphical structures. However the JT is not a transparent representation of the conditional independence as it is in the HJT: in a HJT the domain of its potentials is consistent with the nodes. For example, if $\{a, b, c\}$ is a node in a HJT, the domain of its potential would be $Sp(X_a) \times Sp(X_b) \times Sp(X_c)$, where $Sp(X_v)$ denotes the state space of variable X_v. In a JT a node can contain more elements than the domain of its associated potential.

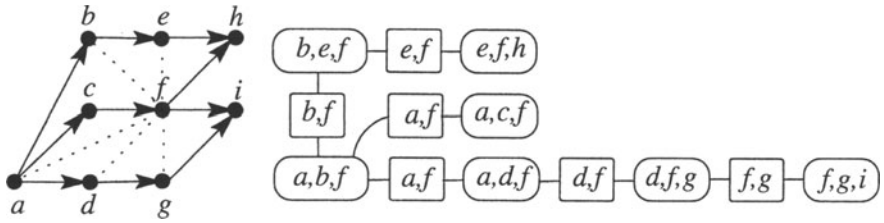
5.4 Propagation

In this section we illustrate how propagation operates in HJTs for a propagation task when we learn a variable in the first layer $I(1)$. If we were to address a propagation task when we learn variables in a higher layer, we would need to first customise the HJT by constructing a JT for the ancestral graph of these conditioning variables. The same propagation principles can then be used for propagation in this customised HJT.

We start with the HJT in Figure 11 constructed from the BN in Figure 10. We compute the propagation task where we learn that $X_a = x'_a$ and we want to compute the distribution of X_h and X_i. At this point the potentials of the non-covering nodes $Fa(v)$ have the conditional distributions $Pr(X_v = x_v | X_{Pa(v)} = x_{Pa(v)})$, except for $Fa(d)$ that has $Pr(X_a = x_a) * P(X_d = x_d | X_a = x_a)$. All separators have been initialised to 1.

Step 1: Propagation within T^1

We enter and propagate that $X_a = x'_a$ using standard JT propagation, described in Section 3. We first choose node $\{a, d\}$ as the root and multiply its potential with the finding for x'_a. We then propagate this information through T^1 by performing a schedule of sum-flows given by a Collect Evidence operation to $\{a, d\}$, followed by a schedule of sum-flows given by a Distribute



(a) **(b)**

Fig. 10. **(a)** The BN in Madsen & Jensen [13]. Edges $\{e, f\}$ and $\{f, g\}$ (*dotted lines*) were added in the moralisation step, and fill-in's $\{a, f\}$, $\{b, f\}$ and $\{d, f\}$ (*dotted lines*) were added in the triangulation step. **(b)** The JT constructed from the BN in (a)

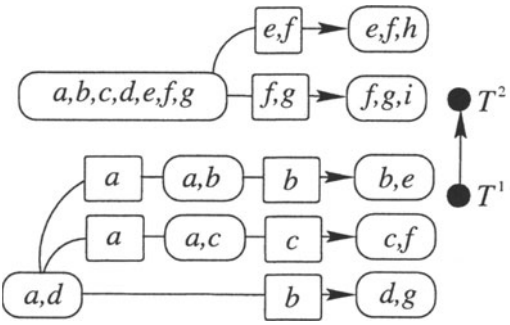


Fig. 11. A HJT for the BN in Figure 10(a)

Evidence operation from $\{a, d\}$. All the sum-flows in the Collect Evidence to $\{a, b\}$ are vacuous because they do not affect any of the potentials. This feature is a consequence of the preservation of the edge direction in the HJT. In the sum-flow from $\{b, e\}$ to $\{a, b\}$, for example, node $\{b, e\}$ has the conditional distribution $Pr(X_e = x_e | X_b = x_b)$ and when we sum-marginalise it, we obtain a potential identical to 1. The potential of the separator $\{b\}$ is already identical to 1. Thus replacing it with the same potential is unnecessary. It is also unnecessary to multiply the potential of node $\{a, b\}$ by 1.

The identification of vacuous sum-flows is also a facility of the LP method [13]. Their method records the edge direction in the potentials which is then used to identify vacuous sum-flows. The LP method is intended for general propagation tasks, that is, the computation of the marginal distribution of any set of nodes conditional of any other set of nodes. This level of generality requires triangulation which does not allow to fully take advantage of the edge direction of the original BN. The sum-flow from node $\{a, c, f\}$ to node $\{a, b, f\}$ is not vacuous because of the fill-in's $\{a, f\}$ and $\{b, f\}$ introduced in the triangulation step in the construction of the JT. In contrast the HJT allows to fully exploit the BN's edge direction and so, none of the sum-

flows in the Collect Evidence to $\{a, d\}$ schedule need to be computed for the propagation task at hand.

We only need to compute the sum-flows in the Distribute Evidence from $\{a, d\}$, and after this schedule has been computed JT T^1 is sum-consistent.

Step 2: Graph simplification

After we have propagated the information through T^1, we can then continue propagating this information to the JT in one layer above, but before this we can remove unnecessary zeros encoded in T^1.

The potential of $\{a, d\}$ is a function that maps (x_a, x_d) to 0 if $x_a \neq x'_a$ and to $Pr(X_a = x'_a, X_d = x_d)$ otherwise. The potential of the separator $\{d\}$ maps x_d to $Pr(X_a = x'_a, X_d = x_d)$. The potential for $\{a, d\}$ is in the numerator of the factorisation encoded in the JT, while the potential for separator $\{d\}$ is in the denominator. These potentials quotient is either zero or one and therefore, we can remove them. The same applies for node $\{a, b\}$ and separator $\{b\}$, and for node $\{a, c\}$ with separator $\{c\}$. The potentials of separators $\{a\}$ are constant functions that take the value $Pr(X_a = x'_a)$ and so, they can be also removed. The resulting JT consist of three disconnected nodes: $\{b, e\}$, $\{c, f\}$ and $\{d, e\}$, displayed in Figure 12. Hugin [10] has a facility for compressing the tables that takes many zeros but the graph of the JT is not simplified by removing the learned nodes, because Hugin is a shell for building BNs for static systems. The HJT adjusts its graphs in line with its potentials, and so, it provides a transparent representation of the factorisation of potentials that it encodes.

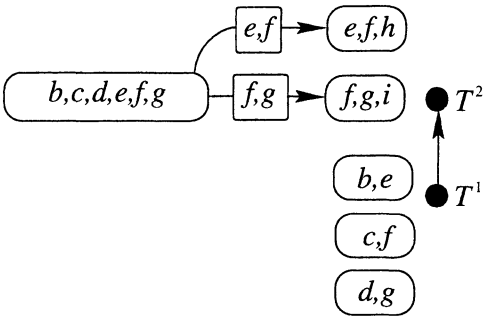


Fig. 12. The HJT after the graph simplification

Step 3: Propagation within T^2

At this point the potentials of the non-covering nodes have the conditional distribution inherited from the original BN, and the separator potentials are

set to 1. The sum-flows in the Collect Evidence schedule to the covering node $\{b, c, d, e, f, g\}$ are all vacuous for the same reasons as we discussed in Step 1. The sum-flows in the Distribute Evidence from $\{b, c, d, e, f, g\}$ are computed. Note that for the sum-flow from $\{b, c, d, e, f, g\}$ to $\{e, f, h\}$, we compute the marginals for X_f and X_e from the disconnected nodes and then we assign this product to the separator $\{e, f\}$. If f and e were in the same JT but not in the same node, we would need to use the method of firing variables for computing their joint distribution, see [5]. The nested junction trees method also uses this type of propagation, [11].

Step 4: Graph Simplification

The purpose of the layers in a HJT is to break cycles. Now that JT T^1 has unconnected nodes, we do not need to have the second layer because there is no cycle to break. We can therefore remove the covering node $\{b, c, d, e, f, g\}$ and connect the upper doors $\{e, f, h\}$ and $\{f, g, i\}$ with the nodes that intersect them. The resulting JT is displayed in Figure 13. The HJT is now sum-consistent and so, the potentials of $\{e, f, h\}$ and $\{f, g, i\}$ now encodes the joint distributions of their associated variables. For example the potential of $\{e, f, h\}$ contains the joint distribution of (X_e, X_f, X_h) from where the marginal distribution of X_h can be computed. Analogously, we can compute the marginal distribution of X_i from the potential of $\{f, g, i\}$.

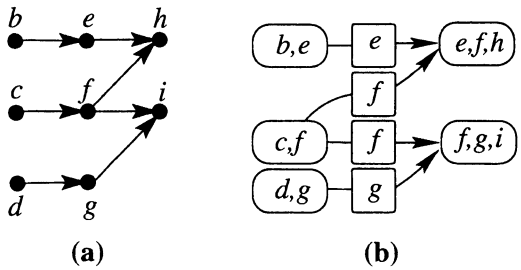


Fig. 13. **(a)** The BN obtained from the BN in Figure 10(a) after learning $X_a = x'_a$ and deleting it. **(b)** The HJT obtained from the HJT in Figure 12 after graph simplification

Notice that if we construct a HJT from the BN in Figure 13(a), we would obtained the HJT in Figure 13(b). This implies that the conditional independence recorded in this BN is the same as the conditional independence in this HJT. This exemplifies the use of incoming conditional independence to simplify the HJT, and so keeping the HJT to the lowest size as possible.

6 Conclusions

We have now defined a formal graphical structure that efficiently codes all conditional independence statements in the BN in a form compatible with the construction of propagation algorithms. It transparently represents the factorisation of the joint distribution of the variables in the system, including features of the factorisation used in the lazy propagation method. This transparency enables us to explore new ways of propagating information for structure inference.

An important building block of the HJT is the ancestrality partition of a BN. This new concept gives a decomposition of a non-decomposable BN into decomposable components. This decomposition facilitated proving the h-separation. We believe that the ancestrality partition may have an important role in providing a framework for proving theorems on BNs.

We have proven with the h-separation theorem that the HJT is constructed with no loss of conditional independence. The HJT construction uses decomposable components of a BN and so, no moralisation and triangulation is required. Propagation tasks only require the moralisation and triangulation of the ancestral graph of the conditioning variables. Total inferential efficiency is achieved when this ancestral graph is decomposable because no conditional independence is lost in this task.

New algorithms can be customised via the HJT to pre-specified inferential tasks and classes of BNs which are not decomposable. In this paper we briefly considered such case. Inferential efficiency gains have important computational cost savings in exact forecasting in dynamic BNs with heterogeneous evolution. We saw that when using JTs for exact forecasting, on-line triangulation is inevitable, even when the lazy propagation method is used because this method operates on a JT. In this dynamic setting, computational savings are achieved with the HJT potentials and graph simplification, which utilises the incoming conditional independence.

We believe that HJTs provide an ideal framework for the development of architectures for probability propagation in highly structured settings.

Acknowledgements

We wish to thank Dr. Lorenz Wernisch for his helpful comments. Part of this research was developed while R. O. Puch was sponsored by Wellcome Trust programme grant GR066790MA.

References

1. SK Andersen, KG Olesen, FV Jensen, and F Jensen. Hugin – a shell for building Bayesian belief universes for expert systems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1080–5. Morgan Kaufmann, San Mateo, California, NS Shridharan (ed.), 1989.

2. RG Cowell, AP Dawid, SL Lauritzen, and DJ Spiegelhalter. *Probabilistic networks and expert systems.* Springer Verlag, 1999.
3. AP Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.
4. Hugin. A shell for building Bayesian networks. *www.hugin.dk*, 2002.
5. FV Jensen. *An introduction to Bayesian networks.* University College London Press, London, United Kingdom, 1996.
6. FV Jensen, SL Lauritzen, and KG Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.
7. K Kanazawa, D Koller, and S Rusell. Stochastic simulation algorithm for dynamic probabilistic networks. pages 346–351. Morgan Kaufmann, San Francisco, California, 1995.
8. U Kjaerulff. Triangulation of graphs – algorithms giving small total state spaces. *Research Report,* Institute of Electronic Systems, Department of Mathematics and Computer Science, Aalborg University, Aalborg, Denmark., 1990.
9. U Kjaerulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 2:7–17, 1992.
10. U Kjaerulff. dHugin: A computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting,* Special Issue on Probability Forecasting, 11:89–111, 1995.
11. U Kjærulff. Inference in Bayesian networks using nested junction trees. In *Learning in graphical models, (Ed. M. I. Jordan)*, pages 51–74. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.
12. P Larrañaga, CMH Kuijpers, M Poza, and RH Murga. Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms. *Statistics and Computing*, 7:19–34, 1997.
13. AL Madsen and FV Jensen. Lazy propagation in junction trees. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (Eds. GF Cooper and S Moral)*, pages 362–9. Morgan Kaufmann, San Francisco, California, 1998.
14. AL Madsen and FV Jensen. Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113:203–245, 1999.
15. RO Puch. *Hierarchical junction trees.* PhD thesis, Department of Statistics, University of Warwick, available at"www.warwick.ac.uk/staff/R.Puch-Solis", 2000.
16. JQ Smith, AE Faria, S French, D Ranyard, D Vlesshhouwer, J Bohunova, T Duranova, M Stubna, L Dutton, C Rojas, and A Sohier. Probabilistic data assimilation within RODOS. *Radiation Proteccion Dosimetry*, 73(1-4):57–9, 1997.
17. JQ Smith and KN Papamichail. Fast Bayes and the dynamic junction forest. *Artificial Intelligence*, 107:99–124, 1999.
18. T Verma and J Pearl. Causal networks: Semantics and expressiveness. In *Uncertainty in Artificial Intelligence 4*, pages 69–76, New York, N. Y., 1988. Elsevier Science Publishing Company, Inc.