A publication of the

Data Analysis Group

Department of Statistics and Applied Mathematics

University of Almería

Technical Report #GAD9702. December 1997.

HUGIN Architecture for Propagating Belief Functions

Antonio Salmerón and Finn V. Jensen

HUGIN Architecture for Propagating Belief Functions †

Antonio Salmerón
Dpt. of Statistics and Applied Mathematics
University of Almería
Spain
e-mail: asc@stat.ualm.es

Finn V. Jensen
Dpt. of Computer Science
Aalborg University
Denmark
e-mail: fvj@cs.auc.dk

Abstract

In this paper we study the use of a HUGIN like architecture for propagating Dempster-Shafer belief functions. The main issue of the proposed scheme is that it allows to do the entire propagation with mass function. To achieve this, an inverse for mass functions is defined.

1 Introduction

Dempster-Shafer Theory of Evidence [5, 13], provides a framework widely used nowadays for dealing with uncertainty. This theory can be interpreted as a more intuitive way of assessing probabilities, assigning masses to subsets of the universe instead of elements of that universe [17].

Belief functions propagation in dependence graphs has been studied in an axiomatic way within the frame of local computation in graphical structures,

†This work has been supported by CICYT under project TIC97-1135-C04-02.

developed by Shafer and Shenoy [14, 15]. Later, Cano, Delgado and Moral [4] expand this scheme introducing some new axioms that allow, for instance, conditioning. Improving the former schemes, Jensen et al. [8] developed an architecture, initially restricted to probabilities, that has been implemented in the expert system shell HUGIN [2]. Recently, Lauritzen and Jensen [11] showed that HUGIN architecture is also valid for Dempster-Shafer belief functions.

In this paper we study the development of a HUGIN like architecture for belief functions, in which the entire propagation, including divisions, are performed over mass functions instead of over commonalities. This scheme is quite close to the fast division architecture by Bissig et al. [3].

The structure under the computations is the *semi lattice representation* (SLR), very similar to *hierarchical trees* [6, 12].

The paper begins reviewing the basic concepts concerning multivariate belief functions in section 2. Some known belief functions propagation methods are described in section 3, while HUGIN architecture with mass functions is studied in section 4. Aspects concerning belief functions representation are treated in section 5, taking as a basis Sandri's hierarchical trees [6, 12]. Calculus with belief functions represented by semi lattices is described in section 6, and the paper ends with conclusions in section 7.

2 Multivariate Belief Functions

This section gives a brief overview of multivariate belief functions, that is, belief functions defined over sets of configurations of variables. For a detailed exposition of this theory, see [5, 13].

Given a variable X we denote by Ω_X its *state space*, i.e. the set of all possible values of X. The elements of a state space Ω_X are called *configurations* of X. In all of this paper, we shall consider finite state spaces. Given a set of variables V, Ω_V denotes the Cartesian product of the state spaces of all the variables in V: $\Omega_V = \prod_{X \in V} \Omega_X$. Assume V_1 and V_2 are sets of variables, $V_1 \subseteq V_2$, and x is a configuration of V_2. Then, $x^{\downarrow V_1}$ is the *projection* of x to V_1, i.e. the configuration of V_1 obtained from x by dropping the coordinates corresponding to variables not in V_1. If $A \subseteq \Omega_{V_2}$, the projection of A to V_1 is defined as $A^{\downarrow V_1} = \{x^{\downarrow V_1} | x \in A\}$. Let V_1, V_2 be sets of variables, $V_1 \subseteq V_2$, and $B \subseteq \Omega_{V_1}$ the *extension* of B to V_2 is defined as its cylinder set: $B^{\uparrow V_2} = B \times \Omega_{V_2 \setminus V_1}$.

Given a set of variables V, a *Dempster-Shafer belief function* is a mapping that assigns a value between 0 and 1 to every subset of Ω_V. For any belief function, the following three representations are equivalent:

Definition 1 *Let V be a set of variables.* A mass function *on V is a mapping $m : 2^{\Omega_V} \to [0,1]$ verifying*

$$m(\emptyset) \quad = \quad 0 \tag{1}$$

$$\sum_{A \subseteq \Omega_V} m(A) \quad = \quad 1 \tag{2}$$

The set $\Gamma \subseteq 2^{\Omega_V}$ *such that* $m(A) \neq 0$ *for all* $A \in \Gamma$ *is called the* support *of the belief function represented by* m*. Each element in* Γ *is called a* focal element*.*

Definition 2 *Let* V *be a set of variables.* A belief function *is a mapping* $Bel : 2^{\Omega_V} \rightarrow [0,1]$ *defined as*

$$Bel(A) = \sum_{B:B \subseteq A} m(B) \quad \forall A \subseteq \Omega_V \tag{3}$$

Definition 3 *Let* V *be a set of variables.* A plausibility function *is a mapping* $Pl : 2^{\Omega_V} \rightarrow [0,1]$ *defined as*

$$Pl(A) = \sum_{B:B \cap A \neq \emptyset} m(B) \quad \forall A \subseteq \Omega_V \tag{4}$$

Definition 4 *Let* V *be a set of variables.* A commonality function *is a mapping* $Q : 2^{\Omega_V} \rightarrow [0,1]$ *defined as*

$$Q(A) = \sum_{B:B \supseteq A} m(B) \quad \forall A \subseteq \Omega_V \tag{5}$$

The conversion from one representation to another is given by the inverse Möbius transform, which is known to be unique for any belief function on a set of variables V:

$$m(A) \quad = \quad \sum_{B:B \supseteq A} (-1)^{|B \setminus A|} Q(B) \tag{6}$$

$$m(A) \quad = \quad \sum_{B:B \subseteq A} (-1)^{|A \setminus B|} Bel(B) \tag{7}$$

From now on we shall talk about belief functions which may be represented as a commonality function, a mass function or whatever. Any of the three representations contains the same information, and is a matter of convenience which one to use. However, mass function representation may be more compact. The extreme case is when $Q(A) = 1$ for all $A \in \Gamma = 2^{\Omega_V}$, which is representable by a mass function with only one focal element, namely, $\Theta = \cup_{A \in \Gamma} A = \Omega_V$, for which $m(\Theta) = 1$.

The information contained in two belief functions m_1 and m_2 defined over a set of variables V can be combined by means of *Dempster's rule of combination* as follows:

$$m_{12}(A) = (m_1 \otimes m_2)(A) = K_{12}^{-1} \sum_{\substack{A_1, A_2 \in \Gamma \\ A_1 \cap A_2 = A}} m_1(A_1) m_2(A_2) \text{ for } \emptyset \neq A \in 2^{\Omega_V} \quad (8)$$

where K_{12} is a normalization constant,

$$K_{12} = \sum_{\substack{A_1, A_2 \in \Gamma \\ A_1 \cap A_2 \neq \emptyset}} m_1(A_1) m_2(A_2) \quad (9)$$

Note that when computing the combination of two masses, new focal elements may appear, but always into the intersection closure of Γ. We shall denote by $\tilde{\Gamma}$ the closure under intersection of Γ.

If we ignore the normalization constant, and denote by Q_m the commonality function corresponding to a mass function m, Dempster's rule can be written in terms of commonalities as follows [11]:

$$(Q_{m_1} \otimes Q_{m_2})(A) = Q_{m_1}(A) Q_{m_2}(A) \text{ for } A \neq \emptyset \quad (10)$$

Now assume we have a set of belief functions $m_1, \ldots, m_n$ defined over the sets of variables $V_1, \ldots, V_n$. These sets are organized in a *junction tree*, that is, an undirected tree where each node is a set of variables and verifying that if a variable is contained in two different nodes, V_i and V_j, then it is also contained in every node in the path connecting V_i and V_j. Every node in a junction tree is called a *cluster*. The universal belief function, m, defined over the set $V = V_1 \cup \cdots \cup V_n$ is the combination of all the former belief functions, i.e. $m = m_1 \otimes \cdots \otimes m_n$. Our goal is to obtain the *marginalization* of this universal belief function to every cluster in the junction tree, namely, $m^{\downarrow V_i}$, $i = 1, \ldots, n$, where

$$m^{\downarrow V_i}(A) = \sum_{\substack{B \subseteq \Omega_V \\ B^{\downarrow V_i} = A}} m(B) \quad \text{for all} \quad A \subseteq \Omega_{V_i} \quad (11)$$

The process of obtaining the marginals of the universal belief function is called *propagation*.

3 Standard Propagation Methods

In all of this section we work with a junction tree $\mathcal{T}$ representing a set of variables $V = V_i \cup \cdots \cup V_n$, each V_i carrying a belief function m_i. An example of junction tree may be found in figure 1. The universal belief function over V will be denoted as $m = m_1 \otimes \cdots \otimes m_n$. The task of computing the marginal of this universal belief function in every set V_i is that the set V may be very large, so

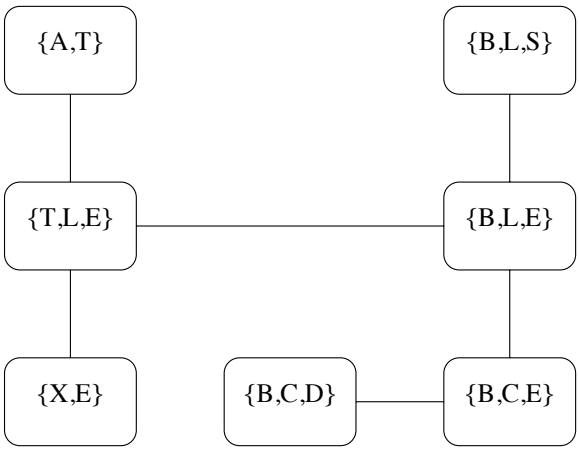

Figure 1: Junction tree for a set of variables $V = \{A, B, C, D, E, L, S, T\}$.

that the straight forward computation of the universal function may be difficult or even impossible, whereas computations concerning with functions over the sets V_i are indeed possible. This last case is known as *local computation*. In this section we point out two different but related architectures for obtaining the marginals of the universal function by local computation. The first one is the Shafer-Shenoy architecture, that has been studied in a very general setting [15], and the other one is the HUGIN architecture, first introduced for probability propagation [8] in the expert system shell HUGIN [2], and recently extended to a more general setting by Lauritzen and Jensen [11]. Both schemes involve successive operations between nodes in the junction tree known as *message passing*. The differences lie either in the form of the messages and in the scheduling of the message passing.

3.1 The Shafer Shenoy architecture

In this scheme, two mailboxes are placed on each edge of the junction tree. Given an edge connecting nodes V_i and V_j, one mailbox is for messages V_i-outgoing and V_j-incoming, and the other mailbox is for the reverse. The messages allocated in both mailboxes will be belief functions defined on $V_i \cap V_j$. Initially, all mailboxes are *empty*, and once a message has been placed on one of them, it is said to be *full*.

A node V_i in a junction tree is allowed to send a message to its neighbour node V_j if and only if all V_i-incoming mailboxes are full except the one from V_j to V_i. Thus, initially only nodes corresponding to leaves can send messages.The message V_i-outgoing and V_j-incoming is computed as

$$\phi_{V_i \to V_j} = \left\{ m_i \otimes \left(\bigotimes_{V_k \in ne(V_i) \setminus V_j} \phi_{V_k \to V_i} \right) \right\}^{\downarrow V_j} \tag{12}$$

where m_i is the initial belief function on V_i, $\phi_{V_k \to V_i}$ are the messages in the mailboxes V_k-outgoing and V_i-incoming and $ne(V_i)$ are the neighbour nodes of V_i. Note that one message contains the information coming from one side of the tree and is sent to the other side of the tree. It can be shown [15] that there is always at least one node allowed to send a message until all mailboxes are full, and when the message passing ends, for every node $V_i \in \mathcal{T}$ it holds that

$$m^{\downarrow V_i} = m_i \otimes \left(\bigotimes_{V_k \in ne(V_i)} \phi_{V_k \to V_i} \right) \tag{13}$$

3.2 The HUGIN architecture

In this case, instead of placing mailboxes on the edges of the junction tree, between every pair of neighbour nodes V_i and V_i, a new node is inserted. Such node is of the form $S_{ij} = V_i \cap V_j$ and is called their *separator*. The set of all separators in $\mathcal{T}$ is denoted by $\mathcal{S}$. Under these conditions, at every moment the universal belief function on V is assumed to factorize as

$$m = \frac{\bigotimes_{i=1}^{n} m_i}{\bigotimes_{S \in \mathcal{S}} m_S} \tag{14}$$

where m_S stands for the belief function defined on S. Initially, the belief functions assigned to the separators are the identity belief function, namely that belief function whose mass is 1 for Ω_V and 0 otherwise, or equivalently, whose commonality is constantly equal to 1. Let this identity be denoted by 1_V. Therefore, in this case expression (14) is clearly the universal belief function on V.

In the HUGIN scheme, the message passing is performed in two steps. A root node is selected in the tree, and then messages are sent starting from the leaves. When a node receives messages from all its neighbour except that one towards the root, it is allowed to send a message upwards, and so on until the root node has received messages from all its neighbours. This is called the COLLECT phase. Now the root node sends a message to all its neighbours, and every node receiving a message itself, sends another one to all of its neighbour except the one from which received the message, and so on until the leaves are reached. This last is called DISTRIBUTE phase. In the HUGIN case, there is only one type of message. Whenever a message is sent from V_i to V_j with separator $S_{ij} = V_i \cap V_j$, the concerned belief functions change as

$$m_i^* = m_i, \quad m_{S_{ij}}^* = m_i^{\downarrow S_{ij}}, \quad m_j^* = m_j \otimes (m_{S_{ij}}^* \otimes m_{S_{ij}}^{-1})^{\uparrow V_j} \tag{15}$$

where $m_{S_{ij}}^{-1}$ stands for the inverse of $m_{S_{ij}}$. It can be seen in the right side of the expression above, that the last combination is extended to V_j. The extension of a mass function is defined as follows: assume a mass function m_i defined over a set of variables $V_i \subseteq V_j$, $A \subseteq \Omega_{V_j}$, then

$$m_i^{\uparrow V_j}(A) = \begin{cases} m_i(B) & \text{if } A \text{ is the cylinder set of } B \subseteq \Omega_{V_i} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Using the same proofs as in [8], it can be shown that after COLLECT and DISTRIBUTE phase, each node and each separator contains the marginal of the universal belief function over V.

4 HUGIN Propagation with Mass Functions

Note that in expression (15) unlike in the Shafer-Shenoy scheme, a division is performed, more precisely, a combination by the inverse of a mass function. This inverse is something the is not already defined, and this section is devoted to find a nice expression of this inverse.

One choice one may make is to use commonality functions instead of mass functions. In this case, there is an easy way of defining the inverse of a Dempster-Shafer belief function, namely, given a set of variables V, $A \subseteq \Omega_V$ and Q a commonality function on V,

$$Q^{-1}(A) = \begin{cases} \dfrac{1}{Q(A)} & \text{if } Q(A) \neq 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

As it is pointed out by Lauritzen and Jensen [11], inverses according to the former expression may result on functions that are not belief functions, since they may lead to negative masses. However, marginalizations are always performed over proper belief functions, and it holds that after a full propagation, and using the inverse defined in (17), the functions on the nodes as well as on the separators of $\mathcal{T}$ are proper belief functions, and, more precisely, they are the marginals of the universal belief function m. This ensures the correctness of HUGIN propagation with belief functions.

Nevertheless, as we pointed out before, mass functions are usually more compact representations than commonalities. We define now an expression for the inverse of a mass function so that there is no need of translating to commonalities before applying the propagation scheme.

Definition 5 *Let m be a mass function over a set of variables W, and with focal elements $\Gamma \subseteq 2^{\Omega_W}$. Let $\tilde{\Gamma}$ be the intersection closure of Γ. We define the* inverse *of m as a function m^{-1} with focal elements in $\tilde{\Gamma}$ with the following expression,*

$$m^{-1}(A) = \begin{cases} \dfrac{1}{\sum\limits_{\substack{B \in \Gamma \\ B \supseteq A}} m(B)} - \sum\limits_{\substack{B \in \tilde{\Gamma} \\ B \supsetneq A}} m^{-1}(B) & \text{If } A \in \tilde{\Gamma} \\ \\ 0 & \text{If } A \notin \tilde{\Gamma} \end{cases} \tag{18}$$

Note that if A is maximal in $\tilde{\Gamma}$, $m^{-1}(A) = 1/m(A)$.

The following two results lead to show that the inverse is well defined. Firstly we must fix some notation. By $Q_{m_1 \cdot m_2}$ we denote the commonality function corresponding to the combination of two masses $m_1 \otimes m_2$, i.e., the product $Q_{m_1} \cdot Q_{m_2}$.

Proposition 1 *Let m be a mass function over a set of variables W, with focal elements Γ, $\tilde{\Gamma}$ the intersection closure of Γ and m^{-1} as in (18). Then,*

$$Q_{m^{-1}}(A) = Q_m^{-1}(A) \quad \forall A \subseteq \Omega_W$$

Proof. We will distinguish three cases.

- If $A \in \tilde{\Gamma}$ is maximal:

  $$Q_{m^{-1}}(A) = \sum_{B \supseteq A} m^{-1}(B) = m^{-1}(A)$$

  and since A is maximal, this is equal to

  $$\frac{1}{m(A)} = \frac{1}{\sum\limits_{B \supseteq A} m(B)} = \frac{1}{Q_m(A)} = Q_m^{-1}(A)$$

  where $B \in \Gamma$.

- If $A \in \tilde{\Gamma}$ is not maximal:

  $$Q_{m^{-1}}(A) = \sum_{B \supseteq A} m^{-1}(B) = m^{-1}(A) + \sum_{B \supsetneq A} m^{-1}(B)$$

  Where $B \in \tilde{\Gamma}$. Substituting the value of $m^{-1}(A)$ according to equation (18), the expression above can be written as follows,

  $$\frac{1}{\sum\limits_{\substack{B \supseteq A \\ B \in \Gamma}} m(B)} - \sum_{\substack{B \supsetneq A \\ B \in \tilde{\Gamma}}} m^{-1}(B) + \sum_{\substack{B \supsetneq A \\ B \in \tilde{\Gamma}}} m^{-1}(B) = Q_m^{-1}(A)$$

- If $A \notin \tilde{\Gamma}$, we must consider two cases:
  1. Assume $\nexists\ B \in \tilde{\Gamma}$ such that $A \subseteq B$. Then,

$$Q_{m^{-1}}(A) = \sum_{B \supseteq A} m^{-1}(B) = \sum_{\substack{B \supseteq A \\ B \notin \tilde{\Gamma}}} m^{-1}(B) = 0$$

since $m^{-1}(B) = 0$ for all $B \notin \tilde{\Gamma}$. Also,

$$Q_m(A) = \sum_{B \supseteq A} m(B) = \sum_{\substack{B \supseteq A \\ B \notin \tilde{\Gamma}}} m(B) = 0 \Rightarrow Q_m^{-1}(A) = 0$$

hence, $0 = Q_{m^{-1}}(A) = Q_m^{-1}(A)$.

2. Assume $\exists C \in \tilde{\Gamma}$ such that $A \subseteq C$. Let be $B = \min\{C \in \tilde{\Gamma} | A \subseteq C\}$. Such a B always exists, because otherwise we could find two minimal sets $B_1, B_2 \in \tilde{\Gamma}$ both containing A and verifying that $B_1 \not\subseteq B_2$ and $B_2 \not\subseteq B_1$ which is equivalent to say that A is contained in $B_1 \cap B_2$ and this is a contradiction with the fact that B_1 and B_2 are minimal. Then,

$$Q_{m^{-1}}(A) = \sum_{\substack{C \supseteq A \\ C \in \tilde{\Gamma}}} m^{-1}(C) = \sum_{\substack{C \supseteq B \\ C \in \tilde{\Gamma}}} m^{-1}(C) = Q_{m^{-1}}(B) = Q_m^{-1}(B)$$

which is true since $m^{-1}(A) = 0$ and $B \in \Gamma$. Now, since $m(A) = 0$, it holds that

$$Q_m(A) = \sum_{C \supseteq A} m(C) = \sum_{C \supseteq B} m(C) = Q_m(B)$$

which implies that $Q_m^{-1}(A) = Q_m^{-1}(B)$, thus, $Q_{m^{-1}}(A) = Q_m^{-1}(B) = Q_m^{-1}(A)$.

■

Theorem 2 *Assume the conditions in proposition 1. Then,*

$$m \otimes m^{-1} = 1_W$$

where 1_W *is the identity belief function on the set of variables* W.

Proof. According to proposition 1, for all $A \in \tilde{\Gamma}$, $Q_{m \cdot m^{-1}}(A) = Q_m(A) \cdot Q_{m^{-1}}(A) = Q_m(A) \cdot Q_m^{-1}(A) = 1 = Q_{1_W}(A) \Rightarrow m \otimes m^{-1} = 1_W$ ■

In this way we have defined the inverse of a mass function, that is not always a mass function. However, as we pointed out before, after a full propagation the results are proper belief functions. The important advance of this approach is that no translations between masses and commonalities are necessary to peform a full propagation.

5 Representation of Belief Functions

The next step when thinking of designing a system for propagating belief functions is to choose an appropriate representation. The key point here is how to represent the focal elements, i.e. the focal sets. In this section we study the classical *bit-string* representation (BSR) and propose a new *sparse* representation, that we call *semilattice* representation (SLR).

5.1 The bit-string representation (BSR)

Consider a set of variables W with finite state space Ω_W. Let $x_1, \ldots, x_n$ be an order of the configurations in Ω_W. Choose any focal element A and define the *bit-string index* of A, with respect to the order above, as

$$I_A = \sum_{x_i \in A} 2^i \tag{19}$$

It can be shown that $0 \leq I_A \leq 2^n$, where $n = |\Omega_W|$. The index 0 corresponds to the empty set and 2^n to Ω_W. Using this scheme, a mass function can be represented as an array indexed by the bit-string indices and storing, in each position of the array, the mass assign to the set corresponding to that position. This representation has many advantages, namely, set operations as intersection and union can be done by fast bit-wise operators, and it is quite easy to move from masses to commonalities and the reverse (see [16]). However, an important restriction arises; this representation is limited to small state spaces. Assume, for instance, that W contains 8 binary variables, then, $|\Omega_W| = 256$ and the number of real numbers required to represent a mass function is 2^{256}, which is a huge number for common computers. However, the number of focal elements of a belief function is usually small with respect to the size of Ω_W (see [1]), what suggests other sorts of representations taking advantage of this fact. In the next section we propose such a representation.

5.2 The semilattice representation (SLR)

In this section we introduce a graphical sparse representation of real valued set functions, thus, valid for any representation of belief functions. The goal is to find a computational scheme where the main operations with quasi belief functions can be done efficiently, namely, combination and inverse as well as other operations necessary to HUGIN architecture, such as extension and marginalization. In this section we are concerned with the definition of the structure.

Definition 6 *Let Ω be a finite set, $f : 2^\Omega \to \mathbb{R}$ a real valued set function. We define the* semilattice representation (SLR) *of function f as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, verifying the following properties:*

1. *The set of nodes $V \subseteq 2^\Omega$ is the set of focal elements of f.*

2. *Attached to each node $X \in V$, there is a real number corresponding to $f(X)$.*

3. *If there is an edge $(X,Y) \in E$, it means that $X \subsetneq Y$ and there is not another $Z \in V$ such that $X \subsetneq Z \subsetneq Y$.*

Example 1 *Let f be a function defined over a space $\Omega = \{a_1, a_2, a_3, a_4\}$, and whose focal elements are $X_1 = \{a_1\}$, $X_2 = \{a_4\}$, $X_3 = \{a_2, a_4\}$, $X_4 = \{a_1, a_2, a_3\}$. Assume the values for those focal elements are $f(X_1) = 0.1$, $f(X_2) = 0.1$, $f(X_3) = 0.2$, $f(X_4) = 0.3$. A SLR for f is displayed in figure 2. Notice that node X_5 is not a focal element (its value is actually 0). It has been added to the diagram when computing the intersection closure of the focal elements.*

Now we want to add a new focal element, say $X_6 = \{a_1, a_3, a_4\}$ with $f(X_6) = 0.3$. We start examining the root nodes in the SLR to find out which of them are subsets of X_6 if any. We find that $X_1, X_2 \subsetneq X_6$. Hence, X_6 must be inserted in the part of the lattice formed by the descendants of both X_1 and X_2, which in this example are none. So, we must connect X_6 to X_1 and X_2, obtaining the diagram in figure 3.

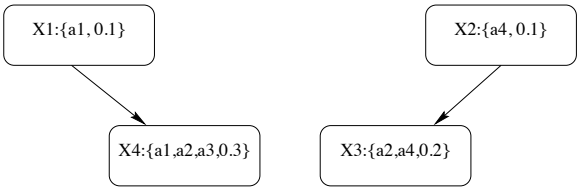

Figure 2: A SLR for f.

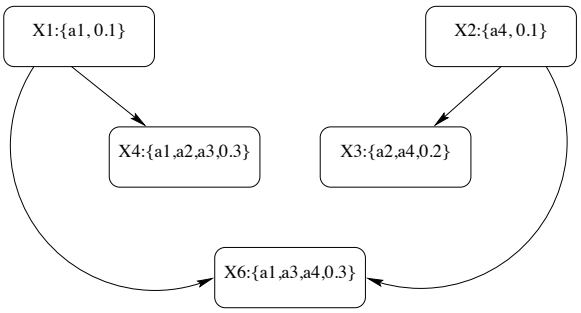

Figure 3: The SLR after inserting X_6.

When constructing the semi lattice, the sets should be inserted in such a way that if a given set X is inserted after any other set Y, then $X \not\subseteq Y$. In this way,

it is avoided to check at every insertion which ones should be the descendants of the new node. To achieve this, it is enough to settle an ordering for the elements to be inserted verifying that for any two sets X_i, X_j, if $|X_i| \leq |X_j|$ then $i \leq j$.

The algorithm below describes the insertion procedure, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the semi lattice, X_i is the set to be added and $f(X_i)$ is the value attached to the set.

INSERT($\mathcal{G}$,X_i,$f(X_i)$)

---

1. Start with an empty set I.
2. For all X_j such that X_j is a root and $X_j \subseteq X_i$, do $I = I \cup \{j\}$.
3. Let H be the set of children of those nodes whose indices are in I and are subsets of X_i. Remove from I those indices corresponding to parents of the nodes in H.
4. If $H \neq \emptyset$ do
   - For each $X_j \in H$, let $I = I \cup \{j\}$.
   - Go to step 3.
5. Add X_i to V and connect X_j to X_i for all $j \in I$.
6. Attach to node X_i the pair $\{X_i, f(X_i)\}$

The set I is intended to contain the indices of the nodes that will be connected to X_i.

In step 2, the algorithm starts exploring the root nodes to find out which of them are subsets of X_i. If any one is found, set I is updated.

Then, in step 3, the idea is to seek within the descendants of the nodes whose indices are in I, those being subsets of X_i.

Notice that when a new set is indexed in I, its parents must be removed from I, to keep condition 3 in definition 6.

Making use of the algorithm above, it is easy to specify the way of constructing the SLR corresponding to any real valued set function with finite support.

Let Ω be a finite set, $f : 2^{\Omega} \to I\!R$ a real valued set function. Let $\Gamma = \{X_1, \ldots, X_n\}$ be the support of f. Assume it holds that $|X_i| \leq |X_j| \Rightarrow i \leq j$. The following algorithm builds a SLR $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for f.

SLR($\mathcal{G}$,f,Γ)

1. For $i = 1$ to n
   (a) **INSERT($\mathcal{G}$,X_i,$f(X_i)$)**

The following example illustrates the construction of a SLR.

Example 2 *Let f be a real valued set function defined over the power set of $\Omega = \{a_1, a_2, a_3, a_4\}$ with focal elements $X_1 = \{a_1\}$, $X_2 = \{a_4\}$, $X_3 = \{a_2, a_4\}$, $X_4 = \{a_1, a_2, a_3\}$, $X_5 = \Omega$. Let the values for these focal elements be $f(X_1) = 0.1$, $F(X_2) = 0.1$, $f(X_3) = 0.2$, $f(X_4) = 0.3$, $f(X_5) = 0.3$. The algorithm starts inserting X_1 and X_2. The insertion is straightforward and no connections have to be made between them. The following set to be inserted is X_3. Then we look at the root nodes, finding X_2 to be the one contained in X_3. Thus, X_3 will be connected to X_2 (see figure 4). In the same fashion, X_4 is added, resulting on the SLR in figure 5. Now we proceed to include X_5. First we explore the root nodes, realizing that both are subsets of X_5. Thus, the exploration is continued from the children of X_1 and X_2, namely, X_3 and X_4. These ones are also subsets of X_5 and, since there are no more nodes to explore, X_5 will be connected to they both. The resulting SLR is displayed in figure 6.*

Figure 4: State of the diagram after inserting X_3.

Now, the SLR must be extended to deal with sets of configurations of variables. Each configuration can be represented as a string of characters. For instance, assume we have two variables X_1 and X_2, and a configuration consisting of X_1 taking its first value and X_2 taking its second value. This configuration can be represented as the string `X101X202`. Instead of repeating a configuration

Figure 5: State of the diagram after inserting X_4.

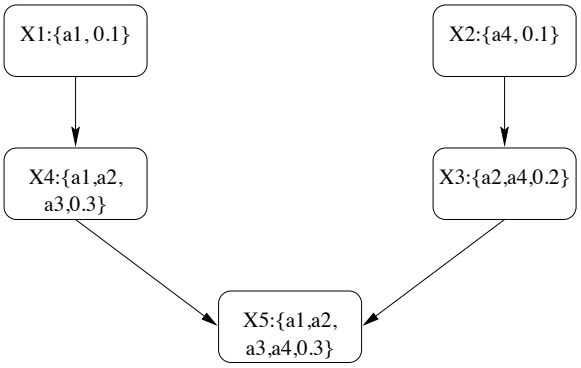

Figure 6: State of the diagram after inserting X_5.

each time it appears in a focal set, all the configurations should be stored in a general table, and then referenced from any set in the system. The general table can be implemented as a *hash table*. From now on we shall denote by $\mathcal{H}$ such table.

The following step is to define how to represent each focal element in a SLR. There are efficient representations of sets, for instance, *balanced binary search trees*, which provides efficient ways of inserting elements and checking for membership, through functions with complexity $O(\log n)$ where n is the number of elements in the set. Each element of the set is just an integer referencing a configuration in $\mathcal{H}$. The correspondence between that number and a configuration is defined as follows: Let N be the number of cells in $\mathcal{H}$, and let x be a configuration stored in the j-th position of the i-th cell. Then, the identifier of that configuration is the integer resulting from concatenating the two numbers i and j resulting in a number ji. Note that if N is close to the total number of configurations in the system, look-up operations in $\mathcal{H}$ are done in a constant time. A detailed explanation of these types of structures can be found in [9].

5.2.1 Computing the Intersection Closure of a SLR

In some cases, it is necessary to add a new feature to the SLR, namely, it must be closed for intersections. The reason is that the inverse of a mass function is defined over the intersection closure of its support, as described in equation (18). Here we explain how to compute the intersection closure of a SLR.

Taking into account the two following facts, the task can be simplified. Firstly, the intersection of a given set with any of its subsets is the subset itself. Secondly, for any two sets A and B, and $C \subseteq B$, it holds that $A \cap C \subseteq A \cap B$ and, furthermore, $A \cap C = (A \cap B) \cap C$.

Then, the strategy to obtain the closure is to divide the nodes in the SLR into different levels. The first one would be the set of all nodes that are leaves (i.e. they have no descendants). Then compute the pairwise intersections for those sets in the first level and insert the new sets into the SLR. The following level would be that consisting of the nodes that are direct predecessors of those in the first level. Again the intersections are computed and the procedure is repeated until there are no more nodes to be explored.

As we said before, whenever a new intersection is calculated, it must be inserted into the SLR. Notice, however, that algorithm **INSERT**, is not applicable since it is designed to insert maximal sets. Here we have to modify it to allow insertions of non maximal sets. Thus, some additional tasks must be carried out. The detailed algorithm is as follows.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a SLR, and $C = A \cap B$ the element to be inserted.

INSERT2($\mathcal{G}$,C,A,B)

1. Start with an empty set I.
2. For all X_j such that X_j is a root and $X_j \subseteq C$, do $I = I \cup \{j\}$.
3. Let H be the set of children of those nodes whose indices are in I, and that are subsets of C. If any of them is equal to C, go to step 7. Otherwise, remove from I those indices corresponding to parents of the nodes in H.
4. If $H \neq \emptyset$ do
   - For each $X_j \in H$, let $I = I \cup \{j\}$.
   - Go to step 3.
5. Add C to $\mathcal{V}$.
6. For each X_k such that $k \in I$, substitute every link $(X_k, Y) \in \mathcal{E}$ such that $C \subset Y$, by the pair of links (X_k, C) and (C, Y).
7. If A is not reachable from C, add the link (C, A).
8. If B is not reachable from C, add the link (C, B).
9. Attach to node C the value 0.0

Now that insertion is defined, it is quite easy to specify an algorithm for computing the intersection closure:

CLOSE($\mathcal{G}$)

1. Let $\mathcal{L} = \{X_1, \ldots, X_n\}$ be the set of leaves in $\mathcal{G}$.
2. Do while $\mathcal{L} \neq \emptyset$
   (a) For $i = 1$ to $n - 1$ do
      i. For $j = i + 1$ to n do

- Let be $X = X_i \cap X_j$.
- If $X \neq \emptyset$, **INSERT2**($\mathcal{G}$,X,X_i,X_j).

(b) Replace $\mathcal{L}$ by the sets of direct predecessors of its elements. Let $\mathcal{L} = \{X_1, \ldots, X_n\}$ be the resulting set. Go to step 2.

Example 3 *Assume the SLR in figure 6. We want to close it for intersections using the algorithm above. We start up from the leaves, taking* $\mathcal{L} = \{X_5\}$. *Since there is only one element in* $\mathcal{L}$ *we go to step 2.(b), obtaining* $\mathcal{L} = \{X_4, X_3\}$. *Now we must compute* $X_4 \cap X_3$ *resulting in* $X_6 = \{a_2\}$. *Inserting this set according to algorithm* **INSERT2** *we obtain the SLR in figure 7. Now we replace* $\mathcal{L}$ *by* $\mathcal{L} = \{X_1, X_2, X_6\}$, *finding out that all the intersections are empty, thus, no new elements must be inserted, and, since there are no direct predecessors of nodes in* $\mathcal{L}$, *the algorithm is finished and the SLR is closed for intersections.*

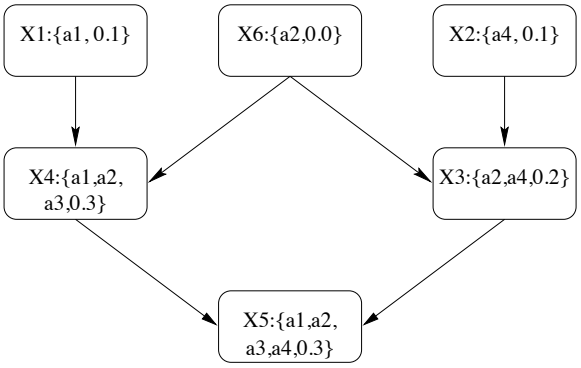

Figure 7: A SLR closed for intersections.

6 Belief Functions Operations on the SLR

Assume the masses are represented by SLRs. We shall describe how the operations in expression (15) should be done over that structure. Namely, these operations are marginalization, inverse, combination and extension.

We start up with **marginalization**. Assume we have a mass function m_V over a set of variables V, represented by a SLR $\mathcal{G}_V$, and we want to compute $m_S = m_V^{\downarrow S}$ for $S \subseteq A$. The procedure is quite simple: for each focal element of m_V, compute its projection to S and update its mass. The following algorithm computes $\mathcal{G}_S$, a SLR for m_S.

MARGINAL($\mathcal{G}_V$,S)

---

1. Select an ancestral order[1] of the sets in the SLR $\mathcal{G}_V$. That is, perform a width first traversal over the lattice. Let $\{A_1, \ldots, A_n\}$ be the resulting order.
2. For $i = 1$ to n
   (a) **INSERT($\mathcal{G}_S$,$A_i^{\downarrow S}$,$m_V(A_i)$)**

Note that an ancestral order always verifies the property required for applying algorithm **INSERT**, that is, if $A_i \subset A_j$ then $i < j$. If that property holds in $\mathcal{G}_V$, it is clear that if $A_i^{\downarrow S} \subset A_j^{\downarrow S}$ then $i < j$, thus, the ordering in $\mathcal{G}_V$ is still valid for $\mathcal{G}_S$.

There is a fact to be considered here. If the set being inserted in $\mathcal{G}_S$ is already an element A_i of $\mathcal{G}_S$ then, instead of assigning the mass $m_V(A_i^{\uparrow V})$, the procedure must be to increase its old mass in an amount of $m_V(A_i^{\uparrow V})$.

The following operation is the **inverse.** We have a mass m_V represented by a SLR $\mathcal{G}_V = (\mathcal{V}_V, \mathcal{E}_V)$, and the goal is to compute m_V^{-1}. According to (18), if Γ is the set of focal elements of m_V, those of m_V^{-1} are also in Γ. Hence, there is no need to construct a new SLR for the inverse, but the old one can be used just updating the masses. But it will be necessary to store two values at each node instead of one: the value of m_V and of m_V^{-1}, in order to facilitate the computations.

Equation (18) shows a way of computing the inverse, in which, for each node, the sum of the inverses and the masses in all the nodes containing it is required. Notice, however, that before computing the inverse, its attached SLR has to be closed for intersections. An algorithmic expression for equation (18) could be as follows.

---

[1] An order of the vertices in a graph is said to be *ancestral* if every node is placed before all its descendants.

INVERSE($\mathcal{G}_V$)

1. **CLOSE($\mathcal{G}_V$).**
2. Select an ancestral order for the sets in $\mathcal{V}_V$. Let $\{A_1, \ldots, A_n\}$ be such order.
3. For $i = n$ downto 1 do
   (a) $a = 0$, $b = m_V(A_i)$.
   (b) For each node $B \in \mathcal{V}_V$ in any path from A_i upwards (i.e. $A_i \subsetneq B$), do
      i. $a = a + m_V^{-1}(B)$
      ii. $b = b + m_V(B)$
   (c) $m_V^{-1}(A_i) = \dfrac{1}{b} - a$

Notice that in this way, when we are going to calculate one inverse, all the required information is known, because we have computed previously the inverse for every superset of the current set in the lattice.

The next operation is **extension**. Assume we have a SLR for a mass function m_i over a set of variables V_i, and we want to obtain a SLR for $m_i^{\uparrow V_j}$, where $V_i \subseteq V_j$. According to equation (16), the focal elements of $m_i^{\uparrow V_j}$ are just the cylinder sets of the focal elements of m_i, that is, for each A focal element of m_i, its cylinder set, $A^{\uparrow V_j}$, will be a focal element of $m_i^{\uparrow V_j}$, and there are no more focal elements. Besides, $m_i^{\uparrow V_j}(A^{\uparrow V_j}) = m_i(A)$. Hence, we can get the SLR for the extension by changing each set in the SLR for m_i by its cylinder set.

Finally, we must define an algorithm for **combination**. More precisely, we want to perform the combination $m_S^* \otimes m_S^{-1}$ in expression (15).

Assume we have two mass functions, m_i and m_j, defined over the sets of variables V. Let $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ be the SLR associated to m_i and $\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j)$ the SLR associated to m_j. The following algorithm computes the SLR $\mathcal{G}_k = (\mathcal{V}_k, \mathcal{E}_k)$ corresponding to $m_k = m_i \otimes m_j$:

COMBINE($\mathcal{G}_i$,$\mathcal{G}_j$)

1. Assume $\mathcal{V}_i = \{A_1, \ldots, A_n\}$ and $\mathcal{V}_j = \{B_1, \ldots, B_m\}$.
2. Compute every nonempty intersection $S = A_i \cup B_j$, $i = 1, \ldots, n$, $j = 1, \ldots, m$ and store it in a list $\mathcal{L}$ together with the value $f(S) = m_i(A_i) \cdot m_j(B_j)$.
3. Sort $\mathcal{L}$ according to the cardinal of the sets on it. Let $\{C_1, \ldots, C_p\}$ be that order.
4. For $i = 1$ to p
   (a) **INSERT($\mathcal{G}_k$,C_i,$f(C_i)$)**. If C_i is already in the SLR $\mathcal{G}_k$, instead of assigning it a mass $f(C_i)$, just update its old mass in an amount of $f(C_i)$.

6.1 Computing Bel and Q on the SLR

In this section we deal with the following task: given a set of variables V, a belief function m_V on V and a SLR $\mathcal{G}_V$, for any set $A \subseteq \Omega_V$, what is the value of $Bel(A)$ and $Q(A)$?

The SLR provides an easy way of answering these questions, just performing graph traversals over the structure. The following algorithm computes $Bel(A)$:

BEL($\mathcal{G}_V$,A)

1. Let $\mathcal{R}$ be the set of nodes in $\mathcal{G}_V$ which are direct predecessors of A.
2. $Bel(A) = m_V(A) + \sum_{B \in \mathcal{R}}$ **bPROP($\mathcal{G}_V$,B,A)**

where **bPROP($\mathcal{G}_V$,B,A)** is a function returning a real value and defined as

bPROP($\mathcal{G}_V$,B,A)

1. $s = m_V(B)$.
2. Mark B as visited.
3. For each node C parent of B in $\mathcal{G}_V$ such that C has not been already visited,
   (a) $s = s$+**bPROP($\mathcal{G}_V$,C,A)**
4. return (s)

The way of computing the commonality of a given set is completely analogous, but instead of starting from the roots, the procedure must start from the leaves and perform a traversal to the roots:

Q($\mathcal{G}_V$,A)

1. Let $\mathcal{L}$ be the set of nodes in $\mathcal{G}_V$ which are direct successors of A.
2. $Q(A) = \sum_{A \subseteq B \in \mathcal{L}}$ **qPROP($\mathcal{G}_V$,B,A)**

where **qPROP($\mathcal{G}_V$,B,A)** is a function returning a real value and defined as

qPROP($\mathcal{G}_V$,B,A)

1. $s = m_V(B)$.
2. Mark B as visited.
3. For each node C child of B in $\mathcal{G}_V$ such that C has not been already visited,

(a) $s = s+\mathbf{qPROP}(\mathcal{G}_V,C,A)$

4. return (s)

7 Conclusions

In this paper we have studied a graphical structure for performing a HUGIN like propagation for Dempster-Shafer belief functions. This structure is based in *hierarchical trees* [6, 12], and adding the feature that it is closed for intersections in some cases where it is necessary (when computing the inverse of a mass function).

Besides, a definition for the inverse of a mass function is proposed. This does not always result in a proper belief function, but it allows to perform an entire propagation with no need of translating into commonalities before computing divisions and then translating back to masses again. It can be shown that, after a full propagation, the result is a proper belief function [11].

References

1. R. Almond (1995) *Graphical belief modeling.* Chapman and Hall, London.

2. S.K. Andersen, K.G. Olesen, F.V. Jensen, F. Jensen (1989) HUGIN- A shell for building Bayesian belief universes for expert systems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 1080-1085. Morgan Kaufmann, San Mateo.

3. R. Bissig, J. Kohlas, N. Lehmann (1997) Fast-division architecture for Dempster-Shafer belief functions. D. Gabbay, R. Kruse, A. Nonnengart and H.J. Ohlbach (eds.), Qualitative and Quantitative Practical Reasoning, First International Joint Conference on Qualitative and Quantitative Practical Reasoning; ECSQARU–FAPR'97. *Lecture Notes in Artificial Intelligence*. Springer.

4. J.E. Cano, M. Delgado, S. Moral (1993) An axiomatic framework for the propagation of uncertainty in directed acyclic networks. *International Journal of Approximate Reasoning* 8, 253-280.

5. A.P. Dempster (1967) Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38, 325-339.

6. V. Dugat, S. Sandri (1994) Complexity of hierarchical trees in Evidence Theory. *ORSA Journal on Computing* 6, 37-49.

7. L.D. Hernández, S. Moral (1997) Inference with idempotent valuations. *Proceedings of the UAI'97 Conference.*

8. F.V. Jensen, S.L. Lauritzen, K.G. Olesen (1990) Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, 4, 269-282.

9. R.L. Kruse, B.P. Leung, C.L. Tondo (1991) *Data structures and program design in C.* Prentice Hall.

10. S.L. Lauritzen, D.J. Spiegelhalter (1988) Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50, 157-224.

11. S.L. Lauritzen, F.V. Jensen (1997) Local computation with valuations from a commutative semigroup. *Annals of Mathematics and Artificial Intelligence* 21, 51-69.

12. S. Sandri (1991) Structuring bodies of evidence. *7th Conference on Uncertainty in AI*, Los Angeles.

13. G.R. Shafer (1976) *A mathematical theory of evidence.* Princeton University Press, Princeton, N.J.

14. G.R. Shafer, P.P. Shenoy (1988) Local computation in hypertrees. Technical Report WP-201, School of Business, University of Kansas.

15. P.P Shenoy, G.R. Shafer (1990) Axioms for probability and belief function propagation. In *Uncertainty in Artificial Intelligence IV*, (ed. R.D. Shachter, T.S. Levitt, L.N. Kanal and J.F. Lemmer), 169-198. North Holland, Amsterdam.

16. H.M. Thoma (1989) Factorization of belief functions. PhD Thesis, Harvard University, Department of Statistics.

17. P. Walley (1991) *Statistical reasoning with imprecise probabilities.* Chapman and Hall.

18. H. Xu (1991) An efficient implementation of belief function propagation. In *Uncertainty in Artificial Intelligence V*, (ed. B.D. D'Ambrosio, Ph. Smets, and P. Bonissone), 425-432. San Mateo, CA: Morgan Kaufmann.

19. H. Xu, R. Kennes (1994) Steps toward efficient implementation of Dempster-Shafer theory. In *Advances in the Dempster-Shafer Theory of Evidence*, (ed. R.R. Yager et al). Wiley.