PROPAGATION OF BELIEF FUNCTIONS:
A DISTRIBUTED APPROACH

Prakash P. Shenoy, Glenn Shafer and Khaled Mellouli

School of Business, University of Kansas, Lawrence, Kansas 66045-2003, USA

I. Abstract and Introduction

In this paper, we describe a scheme for propagating belief functions in certain kinds of
trees using only local computations. This scheme generalizes the computational scheme proposed
by Shafer and Logan\cite{1} for diagnostic trees of the type studied by Gordon and Shortliffe\cite{2,3} and the
slightly more general scheme given by Shafer\cite{4} for hierarchical evidence. It also generalizes the
scheme proposed by Pearl\cite{5} for Bayesian causal trees (see Shenoy and Shafer\cite{6}).

Pearl's causal trees and Gordon and Shortliffe's diagnostic trees are both ways of
breaking the evidence that bears on a large problem down into smaller items of evidence that bear
on smaller parts of the problem so that these smaller problems can be dealt with one at a time. This
localization of effort is often essential in order to make the process of probability judgment feasible,
both for the person who is making probability judgments and for the machine that is combining
them. The basic structure for our scheme is a type of tree that generalizes both Pearl's and Gordon
and Shortliffe's trees. Trees of this general type permit localized computation in Pearl's sense.
They are based on qualitative judgments of conditional independence.

We believe that the scheme we describe here will prove useful in expert systems. It is
now clear that the successful propagation of probabilities or certainty factors in expert systems
requires much more structure than can be provided in a pure production-system framework.
Bayesian schemes, on the other hand, often make unrealistic demands for structure. The
propagation of belief functions in trees and more general networks stands on a middle ground
where some sensible and useful things can be done.

We would like to emphasize that the basic idea of local computation for propagating
probabilities is due to Judea Pearl. It is a very innovative idea; we do not believe that it can be
found in the Bayesian literature prior to Pearl's work. We see our contribution as extending the
usefulness of Pearl's idea by generalizing it from Bayesian probabilities to belief functions.

In the next section, we give a brief introduction to belief functions. The notions of
qualitative independence for partitions and a qualitative Markov tree are introduced in Section III.
Finally, in Section IV, we describe a scheme for propagating belief functions in qualitative Markov
trees.
II. Belief Functions

Suppose \( \Theta \) denotes a set of possible answers to some question, one and only one of which is correct. We call \( \Theta \) a frame of discernment. A function \( \text{Bel} \) that assigns a degree of belief \( \text{Bel}(A) \) to every subset \( A \) of \( \Theta \) is called a belief function if there is a random nonempty subset \( S \) of \( \Theta \) such that \( \text{Bel}(A) = \Pr[S \subseteq A] \) for all \( A \).

Dempster's rule of combination is a rule for calculating a new belief function from two or more belief functions. Consider two random non-empty subsets \( S_1 \) and \( S_2 \). Suppose \( S_1 \) and \( S_2 \) are probabilistically independent, and suppose \( \Pr[S_1 \cap S_2 \neq \emptyset] > 0 \). Let \( S \) be a random non-empty subset that has the probability distribution of \( S_1 \cap S_2 \) conditional on \( S_1 \cap S_2 \neq \emptyset \). If \( \text{Bel}_1 \) and \( \text{Bel}_2 \) are the belief functions corresponding to \( S_1 \) and \( S_2 \), then we denote the belief function corresponding to \( S \) by \( \text{Bel}_1 \oplus \text{Bel}_2 \), and we call \( \text{Bel}_1 \oplus \text{Bel}_2 \) the orthogonal sum of \( \text{Bel}_1 \) and \( \text{Bel}_2 \). The rule for forming \( \text{Bel}_1 \oplus \text{Bel}_2 \) is called Dempster's rule of combination.

Intuitively, \( \text{Bel}_1 \oplus \text{Bel}_2 \) represents the result of pooling the evidence represented by the separate belief functions whenever these items of evidence are independent.

A subset \( S \) of \( \Theta \) is called a focal element of \( \text{Bel} \) if \( \Pr[S = S] \) is positive. In general, combination by Dempster's rule involves the intersection of focal elements. The focal elements for \( \text{Bel}_1 \oplus \ldots \oplus \text{Bel}_n \) will consist of all non-empty intersections of the form \( S_1 \cap \ldots \cap S_n \), where \( S_i \) is a focal element of \( \text{Bel}_i \). The computations involved in combining belief functions by Dempster's rule may become prohibitively complex when \( \Theta \) is large since the number of subsets increases exponentially with the size of the frame. Hence it is important to exploit any special structure in the belief functions being combined that may help us reduce the computational burden.

One case where computational complexity of Dempster's rule can be reduced is the case where the belief functions being combined are "carried" by a partition \( \wp \) of the frame \( \Theta \). The complexity can be reduced in this case because \( \wp \), which has fewer elements than \( \Theta \), can in effect be used in the place of \( \Theta \) when the computations are carried out.

Recall that a set \( \wp \) of subsets of \( \Theta \) is a partition of \( \Theta \) if the sets in \( \wp \) are all non-empty and disjoint, and their union is \( \Theta \). Given a partition \( \wp \) of \( \Theta \), we denote by \( \wp^* \) the set consisting of all unions of elements of \( \wp \); \( \wp^* \) is a field of subsets of \( \Theta \) generated by \( \wp \). We say that a belief function \( \text{Bel} \) over \( \Theta \) is carried by \( \wp \) if the random subset \( S \) corresponding to \( \text{Bel} \) satisfies
Pr[S ∈ ϕ∗] = 1. It is evident that a belief function Bel is carried by the partition ϕ generated by taking intersections of the belief function's focal elements. We can think of such a partition ϕ as a qualitative description of the belief function Bel and will refer to ϕ as the partition associated with Bel.

A partition ϕ of a frame Θ can itself be regarded as a frame. If Bel is a belief function on Θ, then the coarsening of Bel to ϕ is the belief function Belϕ on ϕ given by

Belϕ([P1, ..., Pk]) = Bel(P1 ∪ ... ∪Pk) for every subset \{P1, ..., Pk\} of ϕ. If Bel is a belief function on ϕ, then the vacuous extension of Bel to Θ is the belief function BelΘ given by

BelΘ(A) = Bel(∪{P | P ⊆ A, P ∈ ϕ}). If a belief function is carried by ϕ, then Belϕ contains all the information about Bel. In fact, in this case, Bel can be recovered from Belϕ by vacuous extension: (Belϕ)Θ = Bel. If ϕ1 and ϕ2 are two partitions, and Bel is a belief function on ϕ1, then the projection of Bel to ϕ2 is the result of vacuously extending Bel to Θ and then coarsening to ϕ2.

III. Qualitative Markov Trees

The concept of conditional independence is familiar from probability theory, and it leads within probability theory to many other concepts, including Markov chains and Markov networks. In this section, we introduce a purely qualitative (non-probabilistic) concept of conditional independence and the corresponding concept of a qualitative Markov tree. Qualitative Markov trees are the setting for our computational scheme for propagating belief functions.

Let ϕ1 and ϕ2 be two distinct partitions. We say that ϕ1 is coarser than ϕ2 (or equivalently that ϕ2 is finer than ϕ1), written as ϕ1 ⪯ ϕ2, if for each P2 ∈ ϕ2, there exists P1 ∈ ϕ1 such that P1 ⊇ P2. We call ϕ1 a coarsening of ϕ2 and ϕ2 a refinement of ϕ1.

We write ϕ1 ⪯ ϕ2 to indicate that ϕ1 is coarser than or equal to ϕ2. The relation ⪯ is a partial order and the set of all partitions is a lattice with respect to this partial order (Birkhoff7). The coarsest common refinement of ϕ1, ..., ϕn or the greatest lower bound of ϕ1, ..., ϕn with respect to ⪯, denoted by \∧\{ϕj \ j = 1, ..., n\} or by ϕ1\∧...\∧ϕn, is the partition

\{P1 ∩ ... ∩ Pn \ | \ Pj ∈ ϕj, for j = 1, ..., n, and P1 ∩ ... ∩ Pn ≠ ∅ \}.
We say that $\varphi_1, \ldots, \varphi_n$ are **qualitatively independent**, written as $[\varphi_1, \ldots, \varphi_n] \models \varphi$, if for any $P_j \in \varphi_j$ for $j = 1, \ldots, n$, we have $P_1 \cap \ldots \cap P_n \neq \emptyset$. Furthermore, we say that $\varphi_1, \ldots, \varphi_n$ are **qualitatively conditionally independent** given $\varphi$, written as $[\varphi_1, \ldots, \varphi_n] \models \varphi$, if whenever we select $P \in \varphi$, $P_i \in \varphi_i$ for $i = 1, \ldots, n$ such that

$P \cap P_i \neq \emptyset$ for $i = 1, \ldots, n$, then $P \cap P_1 \cap \ldots \cap P_n \neq \emptyset$. These definitions do not involve probabilities; just logical relations. But (stochastic) conditional independence for random variables does imply qualitative conditional independence for associated partitions (see Shafer, Shenoy and Mellouli\textsuperscript{9} for details).

Qualitative conditional independence is important for belief functions because it is used in defining the circumstances under which we get the right answer when we implement Dempster's rule on a partition rather than on a finer frame (see Shafer\textsuperscript{8}, p. 177).

**Theorem 3.1** If $Bel_1$ and $Bel_2$ are carried by $\varphi_1$ and $\varphi_2$ respectively, and $[\varphi_1, \varphi_2] \models \varphi$, then $(Bel_1 \oplus Bel_2) \varphi = (Bel_1) \varphi \oplus (Bel_2) \varphi$.

Another consequence of qualitative conditional independence is as follows (see Shafer, Shenoy and Mellouli\textsuperscript{9} for details and a proof of this result).

**Theorem 3.2** Suppose that $[\varphi_1, \varphi_2] \models \varphi$. Let $Bel_2$ be carried by $\varphi_2$. Then $(Bel_2) \varphi_1 = ((Bel_2) \varphi) \varphi_1$.

We now consider networks where the nodes represent partitions and the edges represent certain qualitative conditional independence restrictions on the partitions. An (undirected) **network** is a pair $(J, E)$, where $J$, the **nodes** of the network, is a finite set, and $E$, the **edges** of the network, is a set of unordered pairs of distinct elements of $J$. We say that $i \in J$ and $j \in J$ are **adjacent or neighbors** if $(i, j) \in E$. A node is said to be a **leaf** node if it has exactly one neighbor. A network is a **tree** if it is connected and there are no cycles.

**A qualitative Markov network** for $\{\varphi_j | j \in J\}$ is a network $(J, E)$ such that given any three mutually disjoint subsets $J_1$, $J_2$, and $J_3$ of $J$, if $J_1$ and $J_2$ are **separated** by $J_3$ (in the sense that any path from a node in $J_1$ to a node in $J_2$ goes via some node in $J_3$), then $[\forall \{\varphi_j | j \in J_1\}, \forall \{\varphi_j | j \in J_2\}] \models \forall \{\varphi_j | j \in J_3\}$.
If \((J, E)\) is a qualitative markov network for \(\{ \wp_j \mid j \in J \}\) and the network \((J, E)\) is a tree, then we say that \((J, E)\) is a \textit{qualitative Markov tree} for \(\{ \wp_j \mid j \in J \}\). A characterization of qualitative Markov trees is as follows (see Shafer, Shenoy and Mellouli\(^9\) for a proof of this characterization).

**Theorem 3.3** Let \(\{ \wp_j \mid j \in J \}\) be a finite collection of partitions and let \((J, E)\) be a tree. Given any node \(n\) in \(J\), deletion of \(n\) from \(J\) and deletion of all edges incident to \(n\) from \(E\) results in a forest of \(m\) subtrees. Let the collection of nodes in the \(k\)\(^{th}\) subtree be denoted by \(J_{k,n}\). Then \((J, E)\) is a qualitative Markov tree for \(\{ \wp_j \mid j \in J \}\) if and only if for every \(n \in J\),

\[
\left[ \bigwedge \{ \wp_i \mid i \in J_{1,n} \}, \ldots, \bigwedge \{ \wp_i \mid i \in J_{m,n} \} \right] \rightarrow \wp_n.
\]

**IV. Propagating Belief Functions in Qualitative Markov Trees**

Suppose \(T = (J, E)\) is a qualitative Markov tree for \(\{ \wp_i \mid i \in J \}\), and suppose that for every node \(i\) in \(J\) we have a belief function \(\text{Bel}_i\) carried by \(\wp_i\). We are interested in the orthogonal sum of all these belief functions, for which we use the symbol \(\text{Bel}^T:\)

\[
\text{Bel}^T = \oplus \{ \text{Bel}_i \mid i \in J \}.
\]

We do not, however, need to know \(\text{Bel}^T(A)\) for all subsets \(A\) of the frame \(\Theta\). We need to know only \(\text{Bel}^T(A)\) for certain \(A\) that are in the various fields \(\wp_i^*\). This means that we will be satisfied if we can compute the coarsening \(\text{Bel}^T \wp_i^*\) for every node \(i\).

The coarsening \(\text{Bel}^T \wp_i^*\) can in fact be computed efficiently by a simple recursive scheme that begins at the leaf nodes of \(T\) and moves towards node \(i\) while computing belief functions analogous to \(\text{Bel}^T \wp_i^*\) for successively larger subtrees of \(T\). This recursive scheme gains its efficiency from the fact that the computations it requires are local relative to the tree \(T\). In place of a single global application of Dempster's rule, using \(\Theta\) or \(\bigwedge \{ \wp_i \mid i \in J \}\) as our frame, we make many local applications of the rule, using the partitions \(\wp_i^*\) as frames. Since the computational cost of the rule increases exponentially with the size of the frame, these numerous local applications can be inexpensive relative to a global application, provided the \(\wp_i^*\) are all fairly small.
Given a subtree $\mathcal{U} = \{J_\mathcal{U}, E_\mathcal{U}\}$ of $T$, let $\text{Bel}^\mathcal{U}$ denote the orthogonal sum $\bigoplus\{\text{Bel}_i \mid i \in J_\mathcal{U}\}$. Removal of node $n$ (and all edges incident to node $n$) from $T$ results in a set of subtrees, one for each neighbor $k$ of $n$. Let $V_n$ denote the neighbors of $n$, and for each $k$ in $V_n$, let $T_{k,n} = (J_{k,n}, E_{k,n})$ denote the subtree containing $k$ (that results when $n$ is removed from $T$). The basic relation that allows recursive computation of $\text{Bel}^T \varphi_n$ is stated in Theorem 4.1 below.

**Theorem 4.1.** Let $T = (J, E)$ be a qualitative markov tree for $\{\varphi_i \mid i \in J\}$ and let $\text{Bel}_i$ be carried by $\varphi_i$ for each $i$ in $J$. Then

$$\text{Bel}^T \varphi_n = \text{Bel}_n \bigoplus (\bigoplus \{\text{Bel}^{T_{k,n}} \varphi_k \mid k \in V_n\})$$

(4.1)

**Proof:** Since

$$\text{Bel}^T = \text{Bel}_n \bigoplus (\bigoplus \{\text{Bel}^{T_{k,n}} \mid k \in V_n\}),$$

$\text{Bel}_n$ is carried by $\varphi_n$, $\text{Bel}^{T_{k,n}}$ is carried by $\wedge\{\varphi_j \mid j \in J_{k,n}\}$, and

$$[\wedge\{\varphi_j \mid j \in J_{k,n}\}]_{k \in V_n} \rightarrow \varphi_n,$$

it follows from Theorem 3.1 that

$$\text{Bel}^T \varphi_n = \text{Bel}_n \bigoplus (\bigoplus \{\text{Bel}^{T_{k,n}} \varphi_n \mid k \in V_n\}).$$

Since

$$[\varphi_n, \wedge\{\varphi_j \mid j \in J_{k,n}\}] \rightarrow \varphi_k$$

for every $k \in V_n$, it follows from Theorem 3.2 that

$$\text{Bel}^{T_{k,n}} \varphi_n = (\text{Bel}^{T_{k,n}} \varphi_k) \varphi_n.$$  \hspace{1cm} \text{Q.E.D.}$$

The belief functions in the right hand side of (4.1), $\text{Bel}_n$ and $(\text{Bel}^{T_{k,n}} \varphi_k) \varphi_n$ for $k \in V_n$, are all carried by $\varphi_n$, and hence their orthogonal sum can be computed using $\varphi_n$ as a frame. The computation is recursive because $\text{Bel}^{T_{k,n}} \varphi_k$ is the same type of object as $\text{Bel}^T \varphi_n$, except that it is based on the smaller tree $T_{k,n}$. We need, of course, to get the recursion started; we need to be able to compute $\text{Bel}^U \varphi_j$ when $U$ is a tree containing only the node $j$, or perhaps $j$ and some of its neighbors. But this is easy. If $U$ consists of the single node $j$, then (4.1) tells us that

$$\text{Bel}^U \varphi_j = \text{Bel}_j,$$

(4.2)
and if $U$ consists of $j$ and some of its neighbors, say in $V_j$, then (4.1) tells us that

$$
\text{Bel}^U_j \varphi_j = \text{Bel}_j \oplus (\oplus \{ \text{Bel}_k \varphi_j \mid k \in V_j \}).
$$

(4.3)

We can take either (4.2) or (4.3) as the starting point of the recursion.

In order to see more clearly how to direct the recursion, let us introduce some further notation. Given two neighboring nodes $i$ and $j$ in the tree $T$, set

$$
\text{Bel}_{j \rightarrow i} = (\text{Bel}^T_{T_{j,i}} \varphi_j) \varphi_i
$$

where, as noted before, $T_{j,i}$ denotes the subtree containing $j$ that results when $i$ is removed from $T$.

With this notation, (4.1) can be written as

$$
\text{Bel}^T \varphi_n = \text{Bel}_n \oplus (\oplus \{ \text{Bel}_k \rightarrow n \mid k \in V_n \}).
$$

(4.4)

Moreover, Theorem 4.1 applied to $T_{j,i}$ tells us that

$$
\text{Bel}_{j \rightarrow i} = \left( \text{Bel}_j \oplus (\oplus \{ \text{Bel}_k \rightarrow j \mid k \in (V_j \setminus \{i\}) \}) \right) \varphi_i
$$

(4.5)

for any neighboring nodes $i$ and $j$. If $j$ is a leaf node and $i$ is its only neighbor, then the set $V_j \setminus \{i\}$ is empty, and then (4.5) says simply that $\text{Bel}_{j \rightarrow i} = (\text{Bel}_j) \varphi_i$.

Formulae (4.4) and (4.5) suggest a very simple way to program our recursive computations of $\text{Bel}^T \varphi_i$ in a forward chaining production system. We begin with a working memory that contains $\text{Bel}_i$ for each node $i$ in $J$, and we use just two rules:

**Rule 1:**

If $j \in J$, $i \in V_j$, $\text{Bel}_k \rightarrow j$ is present in working memory for every $k$ in $V_j \setminus \{i\}$, and $\text{Bel}_j$ is present in working memory,

then use (4.5) to compute $\text{Bel}_{j \rightarrow i}$ and place it in working memory.

**Rule 2:**

If $i \in J$, $\text{Bel}_k \rightarrow i$ is present in working memory for every $k$ in $V_i$, and $\text{Bel}_i$ is present in working memory,

then use (4.4) to compute $\text{Bel}^T \varphi_i$, and then print it.

Notice that Rule 1 will fire initially only for leaf nodes, since initially no $\text{Bel}_k \rightarrow j$ are in working memory. Rule 1 will eventually fire in both directions for every edge $(i, j)$ producing both $\text{Bel}_{j \rightarrow i}$ and $\text{Bel}_{i \rightarrow j}$. We assume that repetitions of these firings are prevented by a refractory
principle that prevents a rule from firing again for the same instantiation of the antecedent. Rule 2 will eventually fire for every $i$. Thus the total number of firings is equal to $2(|U| - 1) + |U| = 3|U| - 2$.

The potential efficiency of this computational scheme is enhanced by the fact that many of the applications of Dempster's rule on different $\rho_i$ can be carried out in parallel. We can make this potential parallelism graphic by imagining that a separate processor is assigned to each $\rho_i$. The processor assigned to $\rho_i$ computes $\text{Bel}_i^T \rho_i$ and $\text{Bel}_{i \rightarrow k}$ using (4.4) and (4.5) respectively. This means that it combines belief functions using $\rho_i$ as a frame. It also projects belief functions from $\rho_i$ to $\rho_k$, where $k$ is a neighbor of $i$.

Since the processor assigned to $\rho_i$ communicates directly with the processor devoted to $\rho_k$ only when $k$ is a neighbor of $i$, the Markov tree itself can be thought of as a picture of the architecture of the parallel machine; the nodes are processors and the links are communication lines. In this parallel machine, the "working memory" of the production system implementation is replaced by local memory registers at the links. We may assume that every link, there are two sets of memory registers -- one for communication in each direction. Thus at the link between $i$ and $k$, say, there will be one set of registers where $i$ writes $\text{Bel}_{i \rightarrow k}$ for $k$ to read, and another where $k$ writes $\text{Bel}_{k \rightarrow i}$ for $i$ to read. Each processor $i$ also has an input register, where $\text{Bel}_i$ is written from outside the machine, and an output register, where it writes $\text{Bel}^T_i \rho_i$. Figure 1 shows a typical processor, with three neighbors.

We may assume that the processor at $i$ begins work on the computations it is authorized to perform as soon as it receives the necessary inputs. In other words, it computes $\text{Bel}_{i \rightarrow j}$ as soon as it receives $\text{Bel}_i$ and $\text{Bel}_{k \rightarrow i}$ for all $k \in V_i - \{j\}$, and it computes $\text{Bel}^T_i \rho_i$ as soon as it receives $\text{Bel}_i$ and $\text{Bel}_{k \rightarrow i}$ for all $k \in V_i$. If we further assume that the processor does not repeat computations for the same inputs, and if we input all the $\text{Bel}_i$ before turning the processors on, then our parallel machine will operate in fundamentally the same way as the production system described above.
Figure 1 A typical node processor (with three neighbors).
The parallel machine could also be operated in a more dynamic way. Instead of entering all Bel\textsubscript{1} before starting the computations, the Bel\textsubscript{1} could be entered at any time. Initially, to get the computations started, we let all the belief functions Bel\textsubscript{1} to be vacuous. Then as we accumulate independent pieces of evidence, we enter these (non-vacuous) belief functions representing the evidence at the appropriate nodes in the tree. Note that if we have two or more independent pieces of evidence that is represented by belief functions carried by the same node φ\textsubscript{1}, and these belief functions are all entered at φ\textsubscript{1}, then the processor at φ\textsubscript{1} combines all the belief functions input to it to form a belief function Bel\textsubscript{1}. Also note that the refractory principle does not prevent the two rules from firing again for the same edges and nodes if the instantiation is different (as a result of entering a new belief function at a node).

V. Conclusion

The scheme described above is not an algorithm. It does not specify how the coarsenings from one partition to its neighbors are to be carried out, since the most efficient way to do this will depend on the particular nature of the relations between these partitions. The details of the implementation of Dempster's rule at the level of each partition may also depend on the nature of the belief functions being combined there. The general scheme is useful, however, because of its conceptual clarity and its unifying role. In particular, it unifies two computational schemes that had previously seemed rather disparate: Pearl's scheme for propagation of probabilities in Bayesian causal trees (Pearl\textsuperscript{5}) and Shafer and Logan's scheme for combining belief functions in diagnostic trees (Shafer and Logan\textsuperscript{1}). Both these schemes are special cases of the general scheme for propagation in qualitative Markov trees, and they derive most of their computational power from this fact though they also exploit special features of the problem they solve. Pearl's scheme derives some computational power from the simplicity of Bayesian probability measures relative to general belief functions, and Shafer and Logan's scheme derives some computational power from Barnett's technique (Barnett\textsuperscript{10}) which it is able to exploit because the belief functions being combined are "simple support functions" (i.e., have at most two focal elements one of which is the frame Θ). A comparison of Pearl's and Shafer and Logan's schemes with the general scheme presented here is sketched in Shenoy and Shafer\textsuperscript{6}.

VI. Acknowledgements

Research for this paper has been partially supported by NSF grants IST-8405210, IST-8610293, and by ONR grant N00014-85-K-0490.
VII. References


