

An Alternative to Outward Propagation for Dempster-Shafer Belief Functions

Norbert Lehmann and Rolf Haenni

Institute of Informatics, University of Fribourg
1700 Fribourg, Switzerland
norbert.lehmann@unifr.ch

Abstract. Given several Dempster-Shafer belief functions, the framework of valuation networks describes an efficient method for computing the marginal of the combined belief function. The computation is based on a message passing scheme in a Markov tree where after the selection of a root node an inward and an outward propagation can be distinguished. In this paper it will be shown that outward propagation can be replaced by another partial inward propagation. In addition it will also be shown how the efficiency of inward propagation can be improved.

1 Introduction

Dempster-Shafer belief function theory (see [3] and [10]) can be used to represent uncertain knowledge. A piece of evidence is encoded by a belief function. Given many pieces of evidence represented by $\vartheta_1, \dots, \vartheta_m$ and one (or many) hypotheses H , the problem to solve is to marginalize the combined belief function $\vartheta_1 \otimes \dots \otimes \vartheta_m$ in an efficient way to the set of variables of the hypothesis H . The framework of *valuation networks* [12] turns out to be useful for solving this problem because it allows to distribute the belief functions on a *Markov tree*. The computation is then based on a message passing scheme where the two main operations *combination* and *marginalization* are always performed locally on relatively small domains. Depending on the messages which are sent between neighboring nodes of the Markov tree, at least four different architectures can be distinguished:

- the Shenoy-Shafer architecture (see [14]),
- the Lauritzen-Spiegelhalter architecture (see [7] and [8]),
- the HUGIN architecture (see [4] and [5]),
- the Fast-Division architecture (see [2]).

In [16] a comparison of the former three architectures is given for propagating probability functions. The most popular architecture for propagating belief functions is the Shenoy-Shafer architecture. The Lauritzen-Spiegelhalter architecture and especially the HUGIN architecture are more interesting in the field of *Bayesian Networks* [9]. Finally, the Fast-Division architecture is an attempt to apply ideas contained in the HUGIN architecture and the Lauritzen-

Spiegelhalter architecture for propagating belief functions efficiently. In this paper the Shenoy-Shafer architecture will be used. However, the ideas presented here do not depend on the architecture.

For each of these architectures an inward propagation phase and an outward propagation phase can be distinguished. The main contribution of this paper is to show that it is always possible to replace outward propagation by another partial inward propagation.

Furthermore, since only inward propagation is required, it is important to speed up inward propagation. The second contribution of this paper is to develop a heuristic to improve the performance of the inward propagation phase. Intermediate results are stored during the first inward propagation phase such that the number of combinations needed is minimized.

The paper is divided as follows: in Section 2, multivariate Dempster-Shafer belief functions are introduced. Section 3 describes the Shenoy-Shafer architecture. In Section 4, it is shown that instead of an inward propagation followed by an outward propagation, it is sufficient to perform an inward propagation followed by another partial inward propagation. Finally, some improvements to speed up inward propagation are presented in Section 5.

2 Dempster-Shafer Belief Functions

In this section, some basic notions of multivariate Dempster-Shafer belief functions are recalled. More information on the Dempster-Shafer theory of evidence is given in [3], [10], and [15].

Variables and Configurations. We define Θ_x as the *state space* of a variable x , i.e. the set of values of x . It is assumed that all variables have finite state spaces. Upper-case italic letters such as D, E, F, \dots denote sets of variables. Given a set D of variables, let Θ_D denote the Cartesian product $\Theta_D = \times\{\Theta_x : x \in D\}$. Θ_D is called *state space* for D . The elements of Θ_D are *configurations* of D . Upper-case italic letters from the beginning of the alphabet such as A, B, \dots are used to denote sets of configurations.

Projection of Sets of Configurations. If D and D' are sets of variables, $D' \subseteq D$ and \mathbf{x} is a configuration of D , then $\mathbf{x}^{\downarrow D'}$ denotes the *projection* of \mathbf{x} to D' . If A is a subset of Θ_D , then the projection of A to D' , denoted as $A^{\downarrow D'}$, is obtained by projecting each element of A to D' , i.e. $A^{\downarrow D'} = \{\mathbf{x}^{\downarrow D'} : \mathbf{x} \in A\}$.

Extension of Sets of Configurations. If D and D' are sets of variables, $D' \subseteq D$, and B is a subset of $\Theta_{D'}$, then $B^{\uparrow D}$ denotes the *extension* of B to D , i.e. $B^{\uparrow D} = B \times \Theta_{D \setminus D'}$.

2.1 Different Representations

A piece of evidence can be encoded by a belief function. Similar to complex numbers where $c \in \mathbb{C}$ can be represented in polar or rectangular form, there are

also different ways to represent the information contained in a belief function. It can be represented as a *mass function*, as a *belief function*, or as a *commonality function*. In this paper commonality functions are not considered. We therefore focus on (unnormalized) mass functions and belief functions. The unusual notation $[\varphi]_m$ and $[\varphi]_b$ is used instead of m_φ and bel_φ , because in our opinion it is more appropriate and convenient. One of the advantages of this notation is its ability to distinguish easily between normalized and unnormalized mass or belief functions. In accordance with Shafer [11] we speak of *potentials* when no representation specified. We then write only φ , thus without enclosing brackets.

Mass Function. A *mass function* $[\varphi]_m$ on D assigns to every subset A of Θ_D a value in $[0, 1]$, that is $[\varphi]_m : 2^{\Theta_D} \rightarrow [0, 1]$. The following condition must be satisfied:

$$\sum_{A \subseteq \Theta_D} [\varphi(A)]_m = 1. \quad (1)$$

Intuitively, $[\varphi(A)]_m$ is the weight of evidence for A that has not already been assigned to some proper subset of A . Sometimes, a second condition, $[\varphi(\emptyset)]_m = 0$, is imposed. A mass function for which this additional condition holds is called normalized, otherwise it is called unnormalized.

Belief Function. A *belief function* $[\varphi]_b$ on D , $[\varphi]_b : 2^{\Theta_D} \rightarrow [0, 1]$, can be obtained in terms of a mass function:

$$[\varphi(A)]_b = \sum_{B: B \subseteq A} [\varphi(B)]_m. \quad (2)$$

Again, if $[\varphi(\emptyset)]_b = 0$, then the belief function is called normalized. Note that a normalized mass function always leads to a normalized belief function and vice versa.

An unnormalized mass or belief function can always be normalized. We write $[\varphi]_M$ and $[\varphi]_B$ when φ is normalized. The transformation is given as follows:

$$[\varphi(A)]_M = \begin{cases} 0 & \text{if } A = \emptyset, \\ \frac{[\varphi(A)]_m}{1 - [\varphi(\emptyset)]_m} & \text{otherwise.} \end{cases} \quad (3)$$

$$[\varphi(A)]_B = \frac{[\varphi(A)]_b - [\varphi(\emptyset)]_b}{1 - [\varphi(\emptyset)]_b} \quad (4)$$

Given a potential φ on D , D is called the *domain* of φ . The sets $A \subseteq \Theta_D$ for which $[\varphi(A)]_m \neq 0$ are called *focal sets*. We use $FS(\varphi)$ to denote the focal sets of φ .

2.2 Basic Operations

The basic operations for potentials are *combination*, *marginalization*, and *extension*. For each of these operations there are representations which are more appropriate than others. One advantage of the mass function representation is that every basic operation can be performed more or less easily. In this section, we will therefore focus on mass functions.

Combination. Suppose φ_1 and φ_2 are potentials on D_1 and D_2 , respectively. The combination of these two potentials produces an unnormalized potential on $D = D_1 \cup D_2$:

$$[\varphi_1 \otimes \varphi_2(A)]_m = \sum_{B_i \subseteq \Theta_{D_i}} \{[\varphi_1(B_1)]_m \cdot [\varphi_2(B_2)]_m : B_1^{\uparrow D} \cap B_2^{\uparrow D} = A\}. \quad (5)$$

Marginalization. Suppose φ is a potential on D and $D' \subseteq D$. The marginalization of φ to D' produces a potential on D' :

$$[\varphi^{\downarrow D'}(B)]_m = \sum_{A: A^{\downarrow D'} = B} [\varphi(A)]_m. \quad (6)$$

Extension. Suppose φ is a potential on D' and $D' \subseteq D$. The extension of φ to D produces a potential on D :

$$[\varphi^{\uparrow D}(A)]_m = \begin{cases} [\varphi(B)]_m & \text{if } A = B^{\uparrow D}, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

3 The Shenoy-Shafer Architecture

Initially, several potentials $\vartheta_1, \dots, \vartheta_m$ are given. Their domains form a *hypergraph*. For this hypergraph a *covering hypertree* [6] has to be computed first. Then a *Markov tree* [1] can be constructed where $\vartheta_1, \dots, \vartheta_m$ are distributed on the nodes. In such a way every node N_i , $1 \leq i \leq \ell$, contains a potential φ_i on the domain D_i . Each φ_i is equal to the combination of some ϑ_j , and $\vartheta_1 \otimes \dots \otimes \vartheta_m$ is always equal to $\varphi_1 \otimes \dots \otimes \varphi_\ell$.

A Markov tree is the underlying computational structure of the Shenoy-Shafer architecture. Computation is then organized as a message passing scheme where nodes receive and send messages.

3.1 Local Computations

The Dempster-Shafer theory fits perfectly into the framework of *valuation networks* [12]. This framework allows to distribute potentials on a Markov tree and to use the Markov tree afterwards as underlying computational structure.

The framework of valuation networks involves the two operations *marginalization* and *combination*, and three axioms which enable local computation. Potentials satisfy these axioms:

Axiom A1 (Transitivity of marginalization): Suppose φ is a potential on D , and suppose $F \subseteq E \subseteq D$. Then

$$\varphi \downarrow^F = (\varphi \downarrow^E) \downarrow^F. \quad (8)$$

Axiom A2 (Commutativity and associativity): Suppose φ_1 , φ_2 , and φ_3 are potentials on D_1, D_2 , and D_3 , respectively. Then

$$\varphi_1 \otimes \varphi_2 = \varphi_2 \otimes \varphi_1, \quad (9)$$

$$\varphi_1 \otimes (\varphi_2 \otimes \varphi_3) = (\varphi_1 \otimes \varphi_2) \otimes \varphi_3. \quad (10)$$

Axiom A3 (Distributivity of marg. over combination): Suppose φ_1 and φ_2 are potentials on D_1 and D_2 , respectively. Then

$$(\varphi_1 \otimes \varphi_2) \downarrow^{D_1} = \varphi_1 \otimes \varphi_2 \downarrow^{(D_1 \cap D_2)}. \quad (11)$$

Note above that ”=” really means equality and is not only an equivalence relation.

3.2 Message Passing Scheme

Given a Markov tree, computation can be organized as a message passing scheme where nodes receive and send messages to neighbor nodes. A node can send a message to a neighbor node as soon as it has received a message from every other neighbor node. In such a way, a propagation can be started where the leave nodes can compute and send messages first. At the end of the propagation, every node will have received and sent a message from and to each of its neighbor nodes.

If a node N_{k_0} has neighbor nodes N_{k_1}, \dots, N_{k_n} , then the message from N_{k_0} to a neighbor node N_{k_i} is computed as

$$\varphi_{k_0 k_i} = ((\varphi_{k_0} \otimes (\otimes \{\varphi_{k_j k_0} : 1 \leq j \leq n, j \neq i\})) \downarrow^{D_{k_0} \cap D_{k_i}}) \uparrow^{D_{k_i}} \quad (12)$$

As soon as N_{k_0} has received a message from each of its neighbor nodes, it can compute a potential φ'_{k_0} which is equal to the global potential $\varphi := \varphi_1 \otimes \dots \otimes \varphi_\ell$ marginalized to the domain D_{k_0} :

$$\varphi'_{k_0} = \varphi_{k_0} \otimes (\otimes \{\varphi_{k_j k_0} : 1 \leq j \leq n\}). \quad (13)$$

Although the Shenoy-Shafer architecture does not impose to designate a root node, we will nevertheless designate a node N_r as root node. In practice, every node could be selected, but it is better to select a node on which some of the queries can be answered. In such a way an inward propagation phase towards the root node and an outward propagation phase can be distinguished. After inward propagation, the root node is able to compute $\varphi'_r = \varphi \downarrow^{D_r}$. If then an outward propagation is performed, then every other node N_{k_0} can compute $\varphi'_{k_0} = \varphi \downarrow^{D_{k_0}}$. Often, only a partial outward propagation depending on the query is performed. For a set $H \subseteq \Theta_{D_h}$ with $D_h \subseteq D_i$ for a node N_i it is

$$[\varphi \downarrow^{D_h}(H)]_B = [\varphi \downarrow^{D_i}(H \uparrow^{D_i})]_B. \quad (14)$$

Therefore, a node N_i such that $D_h \subseteq D_i$ must be chosen if $[\varphi \downarrow^{D_h}(H)]_B$ has to be computed for a set $H \subseteq \Theta_{D_h}$.

3.3 Binary Join Trees

The Shenoy-Shafer architecture as it is presented above is not very efficient for Markov trees where nodes have more than three neighbors. During outward propagation unnecessary combinations would be recomputed. This problem can be solved by using binary join trees [13] where each node has at most 3 neighbor nodes. Binary join trees minimize the number of combinations needed by saving intermediate results during propagation. If for example a node has n neighbor nodes, it would need $n \cdot (n - 1)$ combinations to generate all messages of this node. In contrast, a corresponding binary join tree would only need $3n - 4$ combinations. Fortunately, every Markov tree can be transformed into a binary join tree by introducing new nodes. For this purpose, for every node with $n > 3$ neighbor nodes it is sufficient to create $n - 3$ new nodes.

4 An Alternative to Outward Propagation

Inward propagation followed by an outward propagation corresponds in a way to a *complete compilation* of the given knowledge allowing then to answer queries of the user very quickly. In the following we propose a method which corresponds only to a *partial compilation* of the given knowledge. This partial compilation results from the inward propagation phase where intermediate results are stored. Later, instead of performing a complete outward propagation phase, a partial inward propagation is performed for every query.

4.1 The Basic Idea

To understand the basic idea of the method presented in this paper, it is better to forget for a while the Shenoy-Shafer architecture. Suppose the potential φ has the domain $d(\varphi) = D$ and suppose $H \subseteq \Theta_D$ is given. Then, according to Equation 4

$$[\varphi(H)]_B = \frac{[\varphi(H)]_b - [\varphi(\emptyset)]_b}{1 - [\varphi(\emptyset)]_b}. \quad (15)$$

As shown by the following theorem, there is another way to compute the same result:

Theorem 1. *Suppose φ has domain $d(\varphi) = D$ and suppose $H \subseteq \Theta_D$ is given. If v is such that $[v(H^c)]_m = 1$ and $d(v) = D$, then*

$$[\varphi(H)]_B = \frac{[(\varphi \otimes v)(\emptyset)]_m - [\varphi(\emptyset)]_m}{1 - [\varphi(\emptyset)]_m}. \quad (16)$$

Proof. It is always $[\varphi(\emptyset)]_b = [\varphi(\emptyset)]_m$. In addition

$$\begin{aligned} [\varphi(H)]_b &= \sum_{A \subseteq H} [\varphi(A)]_m = \sum_{A \cap H^c = \emptyset} [\varphi(A)]_m \\ &= \sum_{A \cap B = \emptyset} ([\varphi(A)]_m \cdot [v(B)]_m) = [(\varphi \otimes v)(\emptyset)]_m. \end{aligned} \quad (17)$$

Therefore, the values $c_1 = [\varphi(\emptyset)]_m$ and $c_2 = [(\varphi \otimes \nu)(\emptyset)]_m$ are sufficient for calculating $[\varphi(H)]_B$.

Our method tries to compute these two values as fast as possible, that's why we use the Shenoy-Shafer architecture. c_1 is computed after the first inward propagation phase. If during inward propagation intermediate results are stored on the nodes, then only a partial inward propagation is needed for c_2 .

4.2 Restrictions on Queries

There is a price we have to pay when the Shenoy-Shafer architecture is used: $[\varphi(H^{\uparrow D})]_B$ (which is equal to $[\varphi^{\downarrow D_h}(H)]_B$) can only be computed for sets $H \subseteq \Theta_{D_h}$ when the underlying Markov tree contains a node N_i with domain D_i such that $D_h \subseteq D_i$. If this is not the case, then a new Markov tree has to be built.

There is a second restriction which may be more severe: Our method is not able to compute queries of the form $[\varphi^{\downarrow D_h}(H)]_m$ and $[\varphi^{\downarrow D_h}(H)]_M$. For a given set $H \subseteq D_h$, our method computes only the value $[\varphi^{\downarrow D_h}(H)]_B$ whereas traditional methods compute the entire mass function $[\varphi^{\downarrow D_h}]_m$.

4.3 Inward Propagation Phase

Prior to the inward propagation phase a root node N_r has to be selected. In practice, every node could be selected, but it is better to select a node on which some of the queries can be answered. Often, the first query is used to determine the root node.

After a root node N_r is designated, an inward propagation towards N_r is started. Every node N_{k_0} with neighbor nodes N_{k_1}, \dots, N_{k_n} where N_{k_n} denotes the node on the path to the root node then computes

$$\tilde{\varphi}_{k_0} = (\varphi_{k_0} \otimes \varphi_{k_1 k_0} \otimes \dots \otimes \varphi_{k_{n-1} k_0}). \quad (18)$$

The message $\varphi_{k_0 k_n}$ is computed by an additional marginalization followed by an extension as

$$\varphi_{k_0 k_n} = \left(\tilde{\varphi}_{k_0}^{\downarrow D_{k_0} \cap D_{k_n}} \right)^{\uparrow D_{k_n}}. \quad (19)$$

The inward propagation phase is finished as soon as N_r has combined his potential with every incoming message yielding φ'_r which is equal to $\varphi^{\downarrow D_r}$. The value $c_1 = [\varphi^{\downarrow D_r}(\emptyset)]_m$ is of particular interest since it is used afterwards for normalization. Therefore, this value has to be stored somewhere.

4.4 Partial Inward Propagation

Suppose for $H \subseteq \Theta_{D_h}$ the query $[\varphi^{\downarrow D_h}(H)]_B$ has to be answered. For this purpose, first a node N_i with domain D_i such that $D_h \subseteq D_i$ has to be selected. If there is no node with this property, then a new Markov tree has to be built. But often there may be several nodes N_i such that $D_h \subseteq D_i$. Then it is best to choose the one which is closest to the root node N_r .

Now, a new potential v with $d(v) = D_h$ and $[v(H^c)]_m = 1$ has to be constructed and adjoined to the node N_i . The idea behind v is derived from propositional logic where for a set of sentences Σ and another sentence h

$$\Sigma \models h \iff \Sigma, \neg h \models \perp. \quad (20)$$

A partial inward propagation is then initiated. Note that only the nodes on the path from N_i to N_r have to perform computations if intermediate results are stored during the first inward propagation phase. The partial inward propagation is finished as soon as the root node N_r has received a message and has computed φ_r'' . It is then

$$\varphi_r'' = (\varphi \otimes v)^{\downarrow D_r}. \quad (21)$$

Of particular interest is the value $[(\varphi \otimes v)^{\downarrow D_r}(\emptyset)]_m$ because it is used to compute $[\varphi^{\downarrow D_h}(H)]_B$. If we let c_1 denote $[\varphi^{\downarrow D_r}(\emptyset)]_m$ obtained from the first inward propagation phase and $c_2 = [(\varphi \otimes v)^{\downarrow D_r}(\emptyset)]_m$ from the partial inward propagation, then the following theorem holds:

Theorem 2. *Suppose $H \subseteq \Theta_{D_h}$ with $D_h \subseteq D_i$ for some node N_i . In addition, suppose that c_1 and c_2 are defined as explained above. Then*

$$[\varphi^{\downarrow D_h}(H)]_B = \frac{c_2 - c_1}{1 - c_1}. \quad (22)$$

Proof. According to Subsection 4.1 and if v is such that $[v(H^c)]_m = 1$ and $d(v) = D_h$, then

$$[\varphi^{\downarrow D_h}(H)]_B = \frac{[(\varphi^{\downarrow D_h} \otimes v)(\emptyset)]_m - [\varphi^{\downarrow D_h}(\emptyset)]_m}{1 - [\varphi^{\downarrow D_h}(\emptyset)]_m} \quad (23)$$

$$= \frac{[(\varphi \otimes v)^{\downarrow D_h}(\emptyset)]_m - [\varphi^{\downarrow D_h}(\emptyset)]_m}{1 - [\varphi^{\downarrow D_h}(\emptyset)]_m} \quad (24)$$

$$= \frac{[(\varphi \otimes v)(\emptyset)]_m - [\varphi(\emptyset)]_m}{1 - [\varphi(\emptyset)]_m} \quad (25)$$

$$= \frac{[(\varphi \otimes v)^{\downarrow D_r}(\emptyset)]_m - [\varphi^{\downarrow D_r}(\emptyset)]_m}{1 - [\varphi^{\downarrow D_r}(\emptyset)]_m} \quad (26)$$

$$= \frac{c_2 - c_1}{1 - c_1} \quad (27)$$

Partial inward propagation is needed to be able to compute $[(\varphi \otimes v)^{\downarrow D_r}(\emptyset)]_m$ on the root node N_r . If during the first inward propagation phase no intermediate results would have been stored, then a complete inward propagation would be necessary to compute $[(\varphi \otimes v)^{\downarrow D_r}(\emptyset)]_m$. Note that partial inward propagation is quite fast since the newly constructed potential v contains only one focal set. Therefore, adjoining v to the node N_i generally simplifies computations.

4.5 The Algorithm

Suppose a Markov tree with nodes N_1, \dots, N_ℓ is given where each node N_i contains a potential φ_i and let $\varphi := \varphi_1 \otimes \dots \otimes \varphi_\ell$ denote the global potential. For each H_k in $\{H_1, \dots, H_m\}$, $H_i \subseteq \Theta_{D_{h_k}}$, $D_{h_k} \subseteq D_i$, the query $[\varphi^{\downarrow D_{h_k}}(H_k)]_B$ has to be answered. The complete algorithm is then given as follows:

Select root node N_r
Inward Propagation : $\varphi'_r = \varphi^{\downarrow D_r}$
Define $c_1 := [\varphi^{\downarrow D_r}(\emptyset)]_m$
If $c_1 = 1$ then **Exit**
for each H_k **in** $\{H_1, \dots, H_m\}$ **do**
 Select node N_i **such that** $D_{h_k} \subseteq D_i$
 Build v **such that** $d(v) = D_{h_k}$ **and** $[v(H_k^c)]_m = 1$
 Adjoin v on node N_i
 Partial Inward Propagation : $\varphi''_r = (\varphi \otimes v)^{\downarrow D_r}$
 Define $c_2 := [(\varphi \otimes v)^{\downarrow D_r}(\emptyset)]_m$
 Output $[\varphi^{\downarrow D_{h_k}}(H_k)]_B = \frac{c_2 - c_1}{1 - c_1}$
Next H_k

Note that if $c_1 = 1$ is obtained then the given knowledge is completely contradictory. In this case there is nothing to do since c_2 would also be equal to 1. Otherwise each query can be answered by a partial inward propagation. Note that when a node N_i has to be selected for a query it is best to choose the node which is closest to the root node N_r .

5 Some Improvements

5.1 Improving the Inward Propagation

In the previous section it has been shown that an inward propagation phase followed by another partial inward propagation is sufficient to answer a query. In order to answer queries as fast as possible it is therefore important that inward propagation does not need too much time. In this subsection an algorithm is presented which minimizes the number of combinations needed by storing intermediate results during the first inward propagation phase. In addition the algorithm tries to determine a good sequence of combinations in order to speed up inward propagation.

Suppose N_{k_0} has neighbor nodes N_{k_1}, \dots, N_{k_n} and suppose N_{k_n} is the node towards the root node (if N_{k_0} is itself the root node, then a new root node connected only to N_{k_0} has to be virtually inserted). During inward propagation N_{k_0} has to compute $\tilde{\varphi}_{k_0} = \varphi_{k_0} \otimes \varphi_{k_1 k_0} \otimes \dots \otimes \varphi_{k_{n-1} k_0}$. Because of associativity and commutativity there are many ways to compute this. Although the final result will always be the same there may be big differences in the time needed. Because the time needed to combine two potentials is correlated to the number of focal sets it seems natural to combine first potentials possessing fewer focal sets. This heuristic is used in the following algorithm:

$\Psi := \{\varphi_{k_0}\} \cup \{\varphi_{k_i k_0} : 1 \leq i < n\}$
loop
 choose $\vartheta_k \in \Psi$ and $\vartheta_l \in \Psi$ **such that**
 $|FS(\vartheta_k)| \leq |FS(\xi)|$ for all $\xi \in \Psi$ **and**
 $|FS(\vartheta_l)| \leq |FS(\xi)|$ for all $\xi \in \Psi \setminus \{\vartheta_k\}$
 $\Psi = \Psi \cup \{\vartheta_k \otimes \vartheta_l\} \setminus \{\vartheta_k, \vartheta_l\}$
until $|\Psi| = 1$

Every node uses this algorithm to combine its incoming messages. At the end the set Ψ contains only the potential $\tilde{\varphi}_{k_0}$. If N_{k_0} is the root node, then $\tilde{\varphi}_{k_0}$ is equal to $\varphi^{\perp D_{k_0}}$, otherwise the message $\varphi_{k_0 k_n}$ to the inward neighbor N_{k_n} can easily be computed using Equation 19 by performing an additional marginalization followed by an extension.

The algorithm can be visualized when for every combination a new node is created. If N_{k_0} has $n > 1$ neighbor nodes, then there are exactly $n - 1$ new nodes with domain D_{k_0} created. Each of these newly created nodes serves to store the corresponding intermediate result. As an example in Figure 1 two trees are shown for a given node with 4 neighbor nodes. The tree on the left corresponds to the computation of $((\varphi_{k_0} \otimes \varphi_{k_1 k_0}) \otimes \varphi_{k_2 k_0}) \otimes \varphi_{k_3 k_0}$ whereas the one on the right corresponds to $((\varphi_{k_0} \otimes \varphi_{k_1 k_0}) \otimes (\varphi_{k_2 k_0} \otimes \varphi_{k_3 k_0}))$.

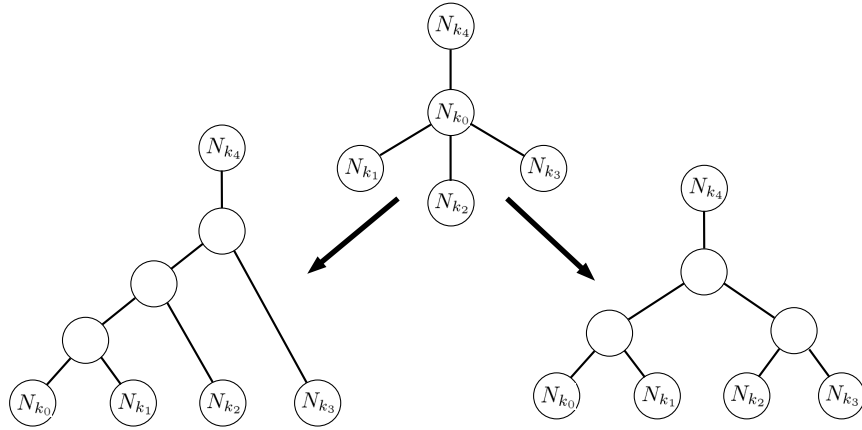


Fig. 1. Visualizing the Algorithm.

The method presented here corresponds to the technique of binary join trees because both methods minimize the number of combinations needed by storing intermediate results during inward propagation. But here the nodes of the Markov tree are not restricted to at most 3 neighbor nodes. But because combination is a binary operation the visualization of the algorithm leads always to a binary join tree.

In [17] another method is presented which minimizes the number of combinations by saving intermediate results during propagation. Every message from the outward neighbors of N_{k_0} are combined accumulatively with the potential of N_{k_0} itself. In Figure 1 the binary join tree on the left where the newly created nodes are on a line corresponds to this method. If n denotes the number of neighbor nodes of N_{k_0} then there are $\prod_{k=2}^{n+1} \binom{k}{2} = \frac{(n+1)! \cdot n!}{2^n}$ possible connections. Only one of them corresponds to the method presented in [17] whereas our method tries to select a good connection depending on φ_{k_0} and the incoming messages $\varphi_{k_i k_0}$.

5.2 Selection of nodes

In the algorithm presented in Subsection 4.5 a node N_i has to be selected on which a newly constructed potential v has to be adjoint. As already mentioned there may be many nodes N_i which could be used for this purpose but it is best to select the one which is closest to the root node N_r . When the heuristic presented in the previous subsection is used during the first inward propagation phase then node N_i has stored some intermediate results, each of them with the domain D_i . To adjoin v it is once again best to select the intermediate result which is closest to the root node. In such a manner the way to the root node is as short as possible.

6 Conclusion

We have shown that given a Markov tree a query can be answered by an inward propagation followed by another partial inward propagation instead of an inward propagation followed by an outward propagation. In order to answer queries as fast as possible it is therefore important to speed up inward propagation. By storing intermediate results during the first inward propagation phase and the use of a heuristic which combines potentials possessing fewer focal sets inward propagation can be improved.

7 Acknowledgements

This work was partially supported by grant No. 21-42927.95 of the Swiss National Foundation for Research.

References

1. R.G. Almond. *Graphical Belief Modeling*. Chapman & Hall, London, UK, 1995.
2. R. Bissig, J. Kohlas, and N. Lehmann. Fast-division architecture for dempster-shafer belief functions. In Dov M.Gabbay, R. Kruse, A. Nonnengart, and H.J. Ohlbach, editors, *First International Joint Conference on Qualitative and Quantitative Practical Reasoning ECSQARU-FAPR'97, Bad Honnef*, pages 198–209. Springer-Verlag, 1997.

3. A.P. Dempster and A.Kong. Uncertain evidence and artificial analysis. *Journal of Statistical Planning and Inference*, 20:355–368, 1988.
4. F.V. Jensen, K.G. Olesen, and S.K. Andersen. An algebra of bayesian belief universes for knowledge-based systems. *Networks*, 20(5):637–659, 1990.
5. F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.
6. J. Kohlas and P.A. Monney. *A Mathematical Theory of Hints. An Approach to the Dempster-Shafer Theory of Evidence*, volume 425 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, 1995.
7. S.L. Lauritzen and F.V. Jensen. Local computation with valuations from a commutative semigroup. *Annals of Mathematics and Artificial Intelligence*, 21(1):51–70, 1997.
8. S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistical Society*, 50(2):157–224, 1988.
9. J. Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.
10. G. Shafer. *The Mathematical Theory of Evidence*. Princeton University Press, 1976.
11. G. Shafer. An axiomatic study of computation in hypertrees. Working Paper 232, School of Business, The University of Kansas, 1991.
12. P.P. Shenoy. A valuation-based language for expert systems. *International Journal of Approximate Reasoning*, 3:383–411, 1989.
13. P.P. Shenoy. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning*, 17:239–263, 1997.
14. P.P. Shenoy and G. Shafer. Axioms for probability and belief functions propagation. In R.D. Shachter and al., editors, *Proceedings of the 4th Conference on Uncertainty in Artificial Intelligence*, pages 169–198. North Holland, 1990.
15. Ph. Smets. Belief functions. In D. Dubois Ph. Smets, A. Mamdani and H. Prade, editors, *Nonstandard logics for automated reasoning*, pages 253–286. Academic Press, 1988.
16. V. Lepar and P.P. Shenoy. A comparison of Lauritzen-Spiegelhalter, Hugin and Shenoy-Shafer architectures for computing marginals of probability distributions. *Uncertainty in Artificial Intelligence*, 14:328–337, 1998.
17. H. Xu. An efficient implementation of belief function propagation. In D’Ambrosio B.D., Smets Ph., and Bonissone P.P., editors, *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 425–432. Morgan Kaufmann, San Mateo, CA, 1997.