Algorithms for Dempster-Shafer Theory

Nic Wilson
School of Computing and Mathematical Sciences
Oxford Brookes University
Gipsy Lane, Headington, Oxford, OX3 0BP, U.K.
pnwilson@brookes.ac.uk

1 Introduction

The method of reasoning with uncertain information known as *Dempster-Shafer theory* arose from the reinterpretation and development of work of Arthur Dempster [Dempster, 67; 68] by Glenn Shafer in his book *a mathematical theory of evidence* [Shafer, 76], and further publications e.g., [Shafer, 81; 90]. More recent variants of Dempster-Shafer theory include the *Transferable Belief Model* see e.g., [Smets, 88; Smets and Kennes, 94] and the *Theory of Hints* e.g., [Kohlas and Monney, 95].

Use of the method involves collecting a number of pieces of uncertain information, which are judged to be 'independent'. Each individual piece of information is represented within the formalism as what is known as a mass function, these are combined using Dempster's rule, and the degrees of belief for various propositions of interest are then calculated. Propositions are expressed as subsets of a set of possibilities, known as the frame of discernment.

Two major problems with the theory are (i) understanding what the calculated values of belief mean; this issue is briefly discussed in section 2.5; and (ii) the computational problems of Dempster's rule, to which most of this chapter is addressed.

The obvious algorithm for calculating the effect of Dempster's rule, as sketched in [Shafer, 76], is (at worst) exponential and [Orponen, 90] has shown that the problem is #P-complete. However Monte-Carlo methods can be used to approximate very closely a value of combined belief, and these have much better complexity: some methods have complexity almost linearly related to the size of the frame, but with a high constant factor because of having to use a large number of trials in order to ensure that the approximation is a good one.

In many applications the frame of discernment is expressed in terms of a product set generated by the values of a number of variables (see section 9), so the size of the frame is itself exponential in the number of variables. Glenn Shafer and Prakash Shenoy have devised techniques for this situation. The exact

methods are again only feasible for a restricted class of problems. Monte-Carlo methods can be used in conjunction with the Shafer-Shenoy techniques, which appear to be more promising.

Section 2 gives the mathematical fundamentals of Dempster-Shafer theory. Section 3 describes algorithms for performing the most important operations on mass functions, and gives their efficiency. Section 4 describes algorithms for converting between the various mathematically equivalent functions used in the theory. Section 5 describes various exact methods for performing Dempster's rule.[1] Some approximate methods are briefly discussed in section 6. Monte-Carlo algorithms are described in section 7. Section 8 shows how other functions used in decision-making can be calculated. Sections 9 and 10 consider algorithms for computing the effect of Dempster's rule for the situation when the frame is a large product set, section 9 giving exact methods, and section 10, Monte-Carlo methods.

2 Some Fundamentals of Dempster-Shafer Theory

This section describes some of the mathematical fundamentals of the theory; most of the material comes from *a mathematical theory of evidence* [Shafer, 76].

2.1 Belief Functions, Mass Functions and Other Associated Functions

Before talking about degrees of belief we must define a set of propositions of interest. To do so we use what Shafer calls a *frame of discernment* (usually abbreviated to *frame*). Mathematically this is just a set, but it is interpreted as a set of mutually exclusive and exhaustive propositions. The propositions of interest are then all assumed to be expressed as subsets of the frame. Throughout this chapter it is assumed that the frame of discernment Θ is finite[2].

If we are considering subsets of a given frame (of discernment) Θ, and $A \subseteq \Theta$ we will sometimes write $\bar{A}$ to mean $\Theta - A$, i.e., the set of elements of Θ not in A.

A *mass function* over Θ (also known as a *basic probability assignment*)[3] is a function $\mathrm{m}\colon 2^\Theta \to [\,0,1\,]$ such that $\mathrm{m}(\emptyset) = 0$ and $\sum_{A \in 2^\Theta} \mathrm{m}(A) = 1$.

[1] Another exact approach is given in chapter 6 of this volume, using probabilistic argumentation systems.

[2] However, some of the Monte-Carlo methods described in section 7 do generalise to infinite frames.

[3] The latter is the term Shafer uses, though he does refer to the values as 'probability masses'. The term 'mass function' is also commonly used, and is a less cumbersome term for such a fundamental entity.

The set of focal sets $\mathcal{F}_{\mathrm{m}}$ of mass function m is defined to be the set of subsets of Θ for which m is non-zero, i.e., $\{A \subseteq \Theta : \mathrm{m}(A) \neq 0\}$. The *core* $\mathcal{C}_{\mathrm{m}}$ of a mass function m is defined to be the union of its focal sets, that is, $\bigcup_{A \in \mathcal{F}_{\mathrm{m}}} A$ (see [Shafer, 76], p40). m can be viewed as being a mass function over $\mathcal{C}_{\mathrm{m}}$, which is sometimes advantageous computationally.

Function $\mathrm{Bel}: 2^{\Theta} \rightarrow [\,0,1\,]$ is said to be a *belief function* over Θ if there exists a mass function m over Θ with, for all $A \in 2^{\Theta}$, $\mathrm{Bel}(A) = \sum_{B \subseteq A} \mathrm{m}(B)$. Bel is said to be the belief function *associated with* m, and will sometimes be written $\mathrm{Bel}_{\mathrm{m}}$.

Clearly, to every mass function over Θ there corresponds (with the above relationship) a unique belief function; conversely for every belief function over Θ there corresponds a unique mass function. To recover mass function m from its associated belief function Bel we can use the following equation ([Shafer, 76], page 39):

$$\text{For } A \subseteq \Theta, \quad \mathrm{m}(A) = \sum_{B \subseteq A} (-1)^{|A-B|} \mathrm{Bel}(B),$$

Belief functions are intended as representations of subjective degrees of belief, as described in [Shafer 76; 81].

The plausibility function[4] $\mathrm{Pl} : 2^{\Theta} \rightarrow [\,0,1\,]$ associated with mass function m is defined by the equations: for all $A \in 2^{\Theta}$, $\mathrm{Pl}(A) = \sum_{B \cap A \neq \emptyset} \mathrm{m}(B)$.

There is a simple relationship between the belief function Bel and the plausibility function Pl associated with a particular mass function m: for $A \subseteq \Theta$, $\mathrm{Pl}(A) = 1 - \mathrm{Bel}(\bar{A})$ and $\mathrm{Bel}(A) = 1 - \mathrm{Pl}(\bar{A})$. The problem of computing values of plausibility is thus equivalent to the problem of computing values of belief; because of this the plausibility function will not be mentioned much in this chapter.

Loosely speaking, a mass function is viewed as a piece of ambiguous evidence that may mean A, for any $A \in \mathcal{F}_{\mathrm{m}}$; we consider that, with probability $\mathrm{m}(A)$, it means A. $\mathrm{Bel}(A)$ can then be thought of as the probability that the ambiguous evidence implies A, and $\mathrm{Pl}(A)$ as the probability that the evidence is consistent with A.

The commonality function $\mathrm{Q} : 2^{\Theta} \rightarrow [\,0,1\,]$ associated with mass function m is defined by the equations: for all $A \in 2^{\Theta}$, $\mathrm{Q}(A) = \sum_{B \supseteq A} \mathrm{m}(B)$. It doesn't usually have a simple interpretation but it allows a simple statement of Dempster's rule (see 2.4). A commonality function determines and is determined by a mass function: if Q_i is the commonality function associated with mass function m_i for $i = 1, 2$, then $\mathrm{Q}_1 = \mathrm{Q}_2$ if and only if $\mathrm{m}_1 = \mathrm{m}_2$.

A very important simple kind of belief function is a simple support function [Shafer, 76]. A simple support function has at most two focal sets, at most one of them being different from the frame Θ. Thus if m is the mass function

[4] [Shafer, 76] uses the term *upper probability function.*

corresponding to a simple support function, then there exists $A \subseteq \Theta$ and $r \in [0,1]$ with $\mathrm{m}(A) = r$ and $\mathrm{m}(\Theta) = 1 - r$. The case where $\mathrm{m}(A) = 1$ represents certain evidence that proposition A is true. Otherwise a simple support function represents uncertain evidence supporting a proposition A.

Simple support functions are fundamental units in *a mathematical theory of evidence* [Shafer, 76]. Dempster's rule also appears to have stronger justifications for the combination of a number of simple support functions (see section 2.5).

2.2 Source Triples

The formalism of [Shafer, 76] was derived from that of Arthur Dempster [Dempster, 67]; Dempster's framework is more convenient for some of the methods for calculating Dempster-Shafer belief, so here we describe his framework (actually a slight variant of it). See also *A Mathematical Theory of Hints* [Kohlas and Monney, 95] which is another Dempster-Shafer style formalism which builds mathematically on Dempster' work.

Define a source triple over Θ to be a triple $(\Omega, \mathrm{P}, \Gamma)$ where Ω is a finite set, P is a strictly positive probability distribution over Ω (so that for all $\omega \in \Omega$, $\mathrm{P}(\omega) \neq 0$) and Γ is a function from Ω to $2^\Theta - \{\emptyset\}$.

One interpretation of source triples is that we're interested in Θ, but we have Bayesian beliefs about Ω, and a logical connection between the two, expressed by Γ. The interpretation of Γ is that if the proposition represented by ω is true, then the proposition represented by $\Gamma(\omega)$ is also true.

Associated with a source triple is a mass function, and hence a belief function and a commonality function, given respectively by $\mathrm{m}(A) = \sum_{\omega\,:\,\Gamma(\omega)=A} \mathrm{P}(\omega)$, $\mathrm{Bel}(A) = \sum_{\omega\,:\,\Gamma(\omega)\subseteq A} \mathrm{P}(\omega)$ and $\mathrm{Q}(A) = \sum_{\omega\,:\,\Gamma(\omega)\supseteq A} \mathrm{P}(\omega)$. Conversely, any mass/belief/commonality function can be expressed in this way for some (non-unique) source triple. For example, if m is a mass function over Θ, we can define a corresponding source triple $(\Omega, \mathrm{P}, \Gamma)$ as follows: define Ω to be $\{1, \ldots, u\}$ where $u = |\mathcal{F}_\mathrm{m}|$, the number of focal sets of m; label the elements of $\mathcal{F}_\mathrm{m}$ as $A_1, \ldots, A_u$ and define P and Γ by, for $j = 1, \ldots, u$, $\mathrm{P}(j) = \mathrm{m}(A_j)$ and $\Gamma(j) = A_j$.

2.3 Conditioning

The conditioned mass function is intended to represent the impact of additional certain information B. Conditioning is a special case of Dempster's rule of combination (see below) and like the general rule requires a sort of independence.

Let m be a mass function over Θ and B be a subset of Θ; also suppose that B has non-empty intersection with the core of m, so that there exists $C \subseteq \Theta$ with $\mathrm{m}(C) \neq 0$ and $C \cap B \neq \emptyset$. Define the mass function m_B, the *conditional*

of m *given* B, by, for $\emptyset \neq A \subseteq \Theta$,

$$\mathrm{m}_B(A) = K \sum_{C:C\cap B=A} \mathrm{m}(C),$$

where normalisation constant K is given by $K^{-1} = \sum_{C:C\cap B\neq\emptyset} \mathrm{m}(C)$.

Let Bel_B be the belief function associated with m_B. It can be calculated directly from Bel, the belief function associated with m, using the equations: for all $A \subseteq \Theta$,

$$\mathrm{Bel}_B(A) = \frac{\mathrm{Bel}(A \cup \bar{B}) - \mathrm{Bel}(\bar{B})}{1 - \mathrm{Bel}(\bar{B})}.$$

Bel_B can be viewed as a belief function over B (rather than Θ), and the same applies to m_B.

We can also condition source triple $(\Omega, \mathrm{P}, \Gamma)$ by B to produce source triple $(\Omega^B, \mathrm{P}^B, \Gamma^B)$ defined as follows: $\Omega^B = \{\omega \in \Omega : \Gamma(\omega) \cap B \neq \emptyset\}$, $\mathrm{P}^B(\omega) = \mathrm{P}(\omega)/\mathrm{P}(\Omega^B)$ for $\omega \in \Omega^B$, and $\Gamma^B(\omega) = \Gamma(\omega) \cap B$. As one would expect, if m is the mass function associated with $(\Omega, \mathrm{P}, \Gamma)$ then m_B is the mass function associated with $(\Omega^B, \mathrm{P}^B, \Gamma^B)$.

2.4 Dempster's Rule of Combination

Suppose we have a number of mass functions (or source triples) each representing a separate piece of information. The combined effect of these, given the appropriate independence assumptions, is calculated using Dempster's rule (of combination). As we shall see, this operation can be computationally very expensive, and this is a major drawback to Dempster-Shafer theory. The major focus of this chapter is computation of combined Dempster-Shafer belief.

2.4.1 Dempster's rule for mass functions

Let m_1 and m_2 be mass functions over frame Θ. Their combination using Dempster's rule, $\mathrm{m}_1 \oplus \mathrm{m}_2$, is defined by, for $\emptyset \neq A \subseteq \Theta$,

$$\mathrm{m}_1 \oplus \mathrm{m}_2(A) = K \sum_{B,C\,:\,B\cap C=A} \mathrm{m}_1(B)\mathrm{m}_2(C),$$

where K is a normalisation constant chosen so that $\mathrm{m}_1 \oplus \mathrm{m}_2$ is a mass function, and so is given by

$$K^{-1} = \sum_{B,C\,:\,B\cap C\neq\emptyset} \mathrm{m}_1(B)\mathrm{m}_2(C).$$

Clearly this combination is only defined when the right hand side of the last equation is non-zero; this happens if and only if the intersection of the cores of m_1 and m_2 is non-empty.

Conditioning can be seen to be mathematically a special case of combination. For $B \subseteq \Theta$, let $\mathcal{I}_B$ be the mass function defined by $\mathcal{I}_B(B) = 1$ (and hence $\mathcal{I}_B(A) = 0$ for $A \neq B$); $\mathcal{I}_B$ expresses certainty that B is true, and is equivalent to asserting proposition B. Conditioning by B is equivalent to combining with $\mathcal{I}_B$: for mass function m, the conditional of m given B, m_B, is equal to $\mathrm{m} \oplus \mathcal{I}_B$.

The operation $\oplus$ is associative and commutative. The combination $\bigoplus_{i=1}^{k} \mathrm{m}_i$ of mass functions $\mathrm{m}_1, \ldots, \mathrm{m}_k$ (over Θ) is well-defined if and only if the intersection of all the cores is non-empty, that is, $\bigcap_{i=1}^{k} \mathcal{C}_{\mathrm{m}_i} \neq \emptyset$. In this case, their combination $\bigoplus_{i=1}^{k} \mathrm{m}_i$ can be shown to be given by, for $\emptyset \neq A \subseteq \Theta$,

$$\bigoplus_{i=1}^{k} \mathrm{m}_i(A) = K_{1,\ldots,k} \sum_{B_1,\ldots,B_k \,:\, B_1 \cap \cdots \cap B_k = A} \mathrm{m}_1(B_1) \cdots \mathrm{m}_k(B_k),$$

where normalisation constant $K_{1,\ldots,k}$ is given by

$$(K_{1,\ldots,k})^{-1} = \sum_{B_1,\ldots,B_k \,:\, B_1 \cap \cdots \cap B_k \neq \emptyset} \mathrm{m}_1(B_1) \cdots \mathrm{m}_k(B_k).$$

The normalisation constant $K_{1,\ldots,k}$ can be viewed as a measure of the conflict between the evidences (see [Shafer, 76, page 65]).

Let $\bigoplus_{i=1}^{k} \mathrm{Bel}_{\mathrm{m}_i}$ and $\bigoplus_{i=1}^{k} \mathrm{Q}_{\mathrm{m}_i}$ be the belief function and commonality function, respectively corresponding to the combined mass function $\bigoplus_{i=1}^{k} \mathrm{m}_i$. They satisfy, for $\emptyset \neq A \subseteq \Theta$,

$$\bigoplus_{i=1}^{k} \mathrm{Bel}_{\mathrm{m}_i}(A) = K_{1,\ldots,k} \sum_{B_1 \ldots, B_k \,:\, \emptyset \neq B_1 \cap \cdots \cap B_k \subseteq A} \mathrm{m}_1(B_1) \cdots \mathrm{m}_k(B_k)$$

and

$$\bigoplus_{i=1}^{k} \mathrm{Q}_{\mathrm{m}_i}(A) = K_{1,\ldots,k}\, \mathrm{Q}_{\mathrm{m}_1}(A) \cdots \mathrm{Q}_{\mathrm{m}_k}(A).$$

This last result can be more succinctly written as $\bigoplus_{i=1}^{k} \mathrm{Q}_{\mathrm{m}_i} = K_{1,\ldots,k} \prod_{i=1}^{k} \mathrm{Q}_{\mathrm{m}_i}$, showing that for commonalities, combination is just normalised product.

The Complexity of Dempster's Rule

[Orponen, 90] has shown that the problem is #P-complete, for calculating a single value of combined belief, mass or commonality, where the problem parameters are $|\Theta|$ and k, respectively the size of the frame of discernment and the number of mass functions being combined. This makes the problem (given the usual assumption of $P \neq NP$) considerably worse than the 'underlying logic' i.e., the operations on subsets (e.g., intersection, complement etc.) which are linear in $|\Theta|$ and k.

2.4.2 Dempster's rule for source triples

The result of applying Dempster's rule to a finite set of source triples $\{(\Omega_i, \mathrm{P}_i, \Gamma_i)$, for $i = 1, \ldots, k\}$, is defined to be the triple $(\Omega, \mathrm{P}_{\mathrm{DS}}, \Gamma)$, which is defined as follows. Let $\Omega^{\times} = \Omega_1 \times \cdots \times \Omega_k$. For $\omega \in \Omega^{\times}$, $\omega(i)$ is defined to be its ith component, so that $\omega = (\omega(1), \ldots, \omega(k))$. Define $\Gamma' \colon \Omega^{\times} \to 2^{\Theta}$ by $\Gamma'(\omega) = \bigcap_{i=1}^{k} \Gamma_i(\omega(i))$ and probability function P' on $\Omega^{\times}$ by $\mathrm{P}'(\omega) = \prod_{i=1}^{k} \mathrm{P}_i(\omega(i))$, for $\omega \in \Omega^{\times}$. Let Ω be the set $\{\omega \in \Omega^{\times} : \Gamma'(\omega) \neq \emptyset\}$, let Γ be Γ' restricted to Ω, and let probability function P_{DS} on Ω be P' conditioned on Ω, so that for $\omega \in \Omega$, $\mathrm{P}_{\mathrm{DS}}(\omega) = \mathrm{P}'(\omega)/\mathrm{P}'(\Omega)$.

The combined measure of belief Bel over Θ is thus given, for $A \subseteq \Theta$, by $\mathrm{Bel}(A) = \mathrm{P}_{\mathrm{DS}}(\{\omega \in \Omega : \Gamma(\omega) \subseteq A\})$, which we abbreviate to $\mathrm{P}_{\mathrm{DS}}(\Gamma(\omega) \subseteq A)$. The mass function associated to the combined source triple is given by $\mathrm{m}(A) = \mathrm{P}_{\mathrm{DS}}(\Gamma(\omega) = A)$. Letting, for $i = 1, \ldots, k$, m_i be the mass function corresponding to $(\Omega_i, \mathrm{P}_i, \Gamma_i)$, then $\mathrm{m} = \mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$ where $\oplus$ is Dempster's rule for mass functions as defined above. Furthermore the normalisation constant $K_{1,\ldots,k}$ defined above equals $1/\mathrm{P}'(\Omega)$.

2.5 Justification of Dempster's Rule

We need to be able to give a meaning to belief/mass functions etc. that is preserved by Dempster's rule. This section is based on material from [Wilson, 93a], which has more discussion of some of the issues.

Dempster's explanation of his rule in [Dempster, 67] amounts to assuming independence (so that for any $\omega \in \Omega^{\times}$, the propositions represented by $\omega(i)$ for $i = 1, \ldots, k$ are considered to be independent) thus generating the product probability function $\mathrm{P}'(\omega) = \prod_{i=1}^{k} \mathrm{P}_i(\omega(i))$, for $\omega \in \Omega^{\times}$. If $\Gamma'(\omega)$ is empty then ω cannot be true; this is because if ω is true then each $\omega(i)$ is true, but then, for each i, $\Gamma_i(\omega(i))$ is true (this is the intended meaning of each Γ_i), so $\emptyset = \Gamma'(\omega) = \bigcap_{i=1}^{k} \Gamma_i(\omega(i))$ is true, but this is impossible as the empty set represents a proposition that cannot be true. Therefore P is then conditioned on the set $\{\omega : \Gamma'(\omega) \neq \emptyset\}$, leading to Dempster's rule.

This two-stage process of firstly assuming independence, and then conditioning on $\Gamma'(\omega)$ being non-empty, needs to be justified. The information given by Γ' is a dependence between ω_i for $i \in \{1, \ldots, k\}$, so they clearly should not be assumed to be independent if this dependence is initially known. Some other justifications also appear not to deal satisfactorily with this crucial point. However, Shafer's random codes canonical examples justification [Shafer, 81, 82b; Shafer and Tversky, 85] does do so.

Shafer's random codes canonical examples

Here the underlying frame Ω is a set of codes. An agent randomly picks a particular code ω with chance $\mathrm{P}(\omega)$ and this code is used to encode a true

statement, which is represented by a subset of some frame Θ. We know the set of codes and the chances of each being picked, but not the particular code picked, so when we receive the encoded message we decode it with each code $\omega' \in \Omega$ in turn to yield a message $\Gamma(\omega')$ (which is a subset of Θ for each ω'). This situation corresponds to a source triple $(\Omega, \mathrm{P}, \Gamma)$ over Θ.

This leads to the desired two-stage process: for if there are a number of agents picking codes stochastically independently and encoding true (but possibly different) messages then the probability distributions are (at this stage) independent. Then if we receive all their messages and decode them we may find certain combinations of codes are incompatible, leading to the second, conditioning, stage.

To represent a piece of evidence, we choose the random codes canonical example (and associated source triple) that is most closely analogous to that piece of evidence. Two pieces of evidences are considered to be independent if we can satisfactorily compare them to the picking of independent random codes. However, in practice, it will often be very hard to say whether our evidences are analogous to random codes, and judging whether these random codes are independent may also be very hard, especially if the comparison is a rather vague one. Other criticisms of this justification are given in the various comments on [Shafer, 82a, 82b], and in [Levi, 83].

Shafer's justification applies only when the underlying probability function has meaning independently of the compatibility function, that is, when the compatibility function is *transitory* [Shafer, 92] (see also [Wilson, 92b] for some discussion of this point). Many occurrences of belief functions are not of this form (in particular, Bayesian belief functions and belief functions close to being Bayesian cannot usually be easily thought of in this way).

[Shafer and Tversky, 85] gives further (and perhaps more natural) canonical examples which only apply to two important special cases: simple support functions (and their combinations) and consonant support functions (a belief function whose focal sets are nested).

Axiomatic justifications of Dempster's rule have also been derived. In [Wilson, 89; 92c] it is shown how apparently very reasonable assumptions determine Dempster's rule for the special case where the input mass functions are simple support functions. [Wilson, 93a] gives a general axiomatic justification; although the assumptions of this general justification seem natural, they are of a somewhat abstract form so it is hard to know in what situations they can safely be made.

There are also justifications of the unnormalised version of Dempster's rule (see section 5) e.g., [Hájek, 92], based on [Smets, 90]; however the meaning of the subsequent values of belief seems somewhat unclear.

There has been much criticism of certain examples of the use of Dempster's rule e.g., [Zadeh, 84; Pearl, 90a, 90b; Walley, 91; Voorbraak, 91; Wilson, 92b].

Many of these criticisms can be countered e.g., those in [Zadeh, 84] were convincingly answered in [Shafer, 84], and [IJAR, 92] has much discussion of Judea Pearl's criticisms. However, despite this, there are situations where Dempster's rule does appear to be counter-intuitive.

It seems to this author that considerable care is needed in the representation of evidence to ensure sensible results.

3 Operations on Mass Functions and Other Mass Potentials

Although we are finally interested in values of belief (Bel), the mass function is in a sense the more fundamental mathematical entity. It is also more convenient for computer storage purposes, at least when the number of focal sets is small. This section considers the efficient implementation of various important operations on mass functions. For convenience, we actually consider a slightly more general object than a mass function.

A *mass potential* m *over* Θ is defined to be a function from 2^Θ to $[\,0,1\,]$. It is said to be *proper* if there exists $\emptyset \neq A \subseteq \Theta$ with $\mathrm{m}(A) \neq 0$.

Mass potentials are a little more general than mass functions, and are used as representations of mass functions: proper mass potential m is intended to represent the mass function m^* given by, $\mathrm{m}^*(\emptyset) = 0$, and for all $\emptyset \neq A \subseteq \Theta$, $\mathrm{m}^*(A) = K\mathrm{m}(A)$ where $K^{-1} = \sum_{A \neq \emptyset} \mathrm{m}(A)$. This operation of mapping m to m^* is called *normalisation*. As for mass functions, the set of focal sets $\mathcal{F}_\mathrm{m}$ of mass potential m is defined to be the set of subsets A such that $\mathrm{m}(A) \neq 0$.

In 3.1 three different ways of representing a mass potential are described, along with indications of how some basic operations are performed with these representations. Section 3.2 describes the operation of conditioning a mass potential; 3.3 briefly discusses calculating the combined core of a number of mass potentials, which can be useful when computing their combination. 3.4 describes lossless coarsening—reducing the size of the frame of discernment without losing information.

3.1 The Representation of a Mass Potential

First we need a representation of subsets of the frame Θ. We enumerate Θ as $\{\theta_1, \ldots, \theta_n\}$. A subset A of Θ will be represented in an n element boolean array with, for $j = 1, \ldots, n$, the jth value in the array being TRUE iff $\theta_j \in A$. Sometimes, to be explicit, this representation of A will be written as `A`; usually, however, the same symbol A will be used for the subset and its representation. This representation `A` may also be thought of as an n-digit binary number, identifying TRUE with 1 and FALSE with 0, and where the most significant

digit represents θ_1. This also defines a total order $<$ on subsets of Θ which we will be using in the second and third representations of mass potentials, with $\emptyset$ being smallest, $\{\theta_n\}$ being next, then $\{\theta_{n-1}\}$, then $\{\theta_{n-1}, \theta_n\}$ and so on, until we reach the largest, Θ.

The complexity of operations on mass potentials depends on their computer representation; different representations have different advantages. Three representations will be discussed.

3.1.1 n-dimensional array representation

Perhaps, the most obvious representation of a mass potential m is in a $|\Theta|$-dimensional array $\{0,1\} \times \cdots \times \{0,1\}$ of reals in $[\,0,1\,]$. The value $\mathrm{m}(\emptyset)$ is put in position $(0,\ldots,0,0)$, the value $\mathrm{m}(\{\theta_n\})$ is put in position $(0,\ldots,0,1)$ and so on, so that the value $\mathrm{m}(A)$ is put in position corresponding to $\mathtt{A}$.

With this representation looking up a value $\mathrm{m}(A)$ (for given A) is fast: time proportional to $|\Theta|$. However, many operations take time exponential in $|\Theta|$, such as normalising, listing the focal sets and conditioning. If the number of focal sets $|\mathcal{F}_\mathrm{m}|$ is close to $2^{|\Theta|}$, this is inevitable; however very often in problems of interest, $|\mathcal{F}_\mathrm{m}|$ is very much smaller than $2^{|\Theta|}$, so it makes sense to use a representation which makes use of this sparseness.

3.1.2 Ordered list representation

This representation of a mass potential is a list of all pairs $(\mathtt{A}, r)$, where $A \in \mathcal{F}$, the set of focal sets of m, and $r = \mathrm{m}(A) \neq 0$. Each $A \in \mathcal{F}$ may appear only once so the length of the list is $|\mathcal{F}|$. Furthermore, the ordering of this list is determined by the sets A, using the total (lexicographic) ordering of subsets given above with smallest A first. Hence if $\mathrm{m}(\emptyset) \neq 0$, the first element in the list is $\big(\emptyset, \mathrm{m}(\emptyset)\big)$. We can write the list as $[(A_1, r_1), \ldots, (A_u, r_u)]$ where $u = |\mathcal{F}|$ and for $t = 1, \ldots, u-1$, $A_t < A_{t+1}$.

It is important (at least for some of the operations) that the computer representation of the list allows 'random access', that is, we can move quickly to any pair (A_t, r_t), without having to work our way through the earlier pairs; for example, we might use a 1-dimensional array to represent the list. We will assume[5] that, given t, the time to retrieve r_t takes $\log |\mathcal{F}|$, retrieving a single

[5]It might be suggested that we're being too conservative here, and instead retrieving r_t and a single co-ordinate of A_t should be assumed to take just constant time. However, at least in an idealised computer, it seems appropriate to assume that the time to access an element of an array of size u does depend on u, and logarithmically because it needs proportional to $\log u$ bits to store the address, which will need to be processed in the retrieval.

It might even be argued that there's actually a very small $u^{1/3}$ term because of the travelling time of the information: each element of the array needs a certain amount of (physical) space to store it, space is 3-dimensional, and the speed of travel is limited by the speed of light. However the array, and hence the computer, would probably have to be of (literally) astronomical size for this term to be significant.

co-ordinate of A_t (if we want to find out if $\theta_j \in A_t$) takes $\log|\mathcal{F}| + \log|\Theta|$ and retrieving the whole of A_t takes time proportional to $|\Theta|$ (since $\log_2|\mathcal{F}| \leq |\Theta|$).

With this representation normalisation can be performed in time proportional to $|\mathcal{F}|$. Listing the focal sets can be also done quickly: in time proportional to $|\mathcal{F}| \times |\Theta|$.

Looking up a value $\mathrm{m}(A)$ (for given A) is more complicated than for the n-dimensional array representation. We can use a binary search method to check if (A, r) is in the list: we first look at A_t for t closest to $u/2$ and see if $A < A_t$; if so we look at $A_{t'}$ for t' closest to $t/2$; otherwise, we look at $A_{t''}$ for t'' closest to $3t/2$ etc. The number of steps is proportional to $\log|\mathcal{F}|$. Each step involves retrieving a set and checking the relative ordering of two subsets, which may be performed in time proportional to $|\Theta|$. Thus the computation[6] is at worst proportional to $|\Theta|\log|\mathcal{F}|$.

An obvious algorithm can calculate $\mathrm{Bel}(A)$, for a given set A, in time proportional to $|\mathcal{F}| \times |\Theta|$, though again, in certain circumstances it may well be possible to perform this operation faster.

Sorting This is a crucial issue for this representation, as it is needed when we condition or combine. Suppose we have an unordered list of pairs (A, r) of length v, possibly with some sets A appearing more than once. How do we sort them into the desired ordered list representation, with no repeats? This ordered list should consist of pairs (A, r'), where r' is the sum of all r such that (A, r) appears in the unordered list.

An algorithm similar to QuickSort may be used:

Split the unordered list into two lists, the first consisting of pairs (A, r) with $A \not\ni \theta_1$, the second consisting of pairs (A, r) with $A \ni \theta_1$.

We will recursively sort each of the two lists and then append the two sorted lists.

If the first list is empty we need do nothing more with it. Otherwise, to sort the first list we split the list into two lists, one consisting of pairs (A, r) with $A \not\ni \theta_2$.

This downward recursion proceeds until we have gone through all θ_j, $j = 1, \ldots, n$. At this point, we always generate a list such that if (A, r) and (A', r') are elements of this list then we must have $A = A'$. Therefore we can merge all these terms creating a single element list $[(A, r'')]$ with r'' being the sum of all the r in the list.

There are $|\Theta|$ levels in the recursion, and the total number of operations needed for each level is proportional to $v(\log v + \log|\Theta|)$ so the number of operations required is proportional to $|\Theta|v(\log v + \log|\Theta|)$.

[6] It may well be possible to improve on this, as we'll usually only need to look at a small number of co-ordinates to check the relative ordering of A and some A_t, since we'll usually know the initial co-ordinates from earlier steps. It thus may be that looking up a value $\mathrm{m}(A)$ can on average be done in time close to $|\Theta|$.

3.1.3 Binary Tree representation

The above sorting procedure suggests a refinement of the ordered list representation: a binary tree whose leaves are pairs $(A, \mathrm{m}(A))$ for focal sets A, and whose branches and other nodes allow quick access to the focal sets. Each split in the tree divides focal sets containing a particular θ_j from those focal sets not containing θ_j.

More precisely: We first find smallest k such that $\{A \in \mathcal{F} : A \not\ni \theta_k\}$ and $\{A \in \mathcal{F} : A \ni \theta_k\}$ are both non-empty. Call these two sets $\mathcal{F}_0$ and $\mathcal{F}_1$ respectively. The first $k-1$ digits of all $A \in \mathcal{F}$ (viewed as $|\Theta|$-digit binary numbers) are the same; call this $k-1$ digit number b. Label the root node F_b^{k-1}.

We will imagine the tree being drawn with the root at the top, going down to the leaves at the bottom.

Create two branches, labelled 0 and 1, going down from the root node. In the tree we will construct, the leaves below the 0 branch will be all pairs $(A, \mathrm{m}(A))$ with $A \in \mathcal{F}_0$. The binary representation of such A all begin with $b0$.

Proceeding recursively, we find smallest l (if one exists) such that $\{A \in \mathcal{F}_0 : A \not\ni \theta_l\}$ and $\{A \in \mathcal{F}_0 : A \ni \theta_l\}$ are both non-empty. The first $l-1$ digits of all $A \in \mathcal{F}_0$ are the same; call this $l-1$ digit number b'. Label the node at the end of the 0 branch $F_{b'}^{l-1}$.

On the other hand, if no such l exists, then $\mathcal{F}_0$ has just one element, call it A. We then create leaf node $(A, \mathrm{m}(A))$ at the end of the 0 branch (with, as ever, set A being represent by a boolean array of size $|\Theta|$).

The same procedure is used at each non-leaf node to create two new nodes.

The total number of nodes in the tree is $2|\mathcal{F}|-1$, so the storage space required is proportional to $|\Theta||\mathcal{F}|$. This construction is very closely related to the algorithm given for sorting a list representation, given above. The time required to construct the binary tree in this way is proportional to $|\Theta||\mathcal{F}|(\log|\Theta| + \log|\mathcal{F}|)$.

The time necessary to insert a new leaf node into the binary tree is proportional to $|\Theta|$ (a new internal node is added in the process). We could also construct the tree by starting off with just one focal set, constructing the associated binary tree (which has just one node), and then incrementally insert the other focal sets. Perhaps surprisingly, this way of constructing the binary tree can be a little more efficient, taking time proportional to $|\Theta||\mathcal{F}|$. This incremental construction is also useful when combining mass functions.

Because of the overheads of this representation, some operations, such as normalisation and listing all the focal sets, take slightly longer (by a constant factor) than the ordered list representation. Other operations, however, are faster with this representation; for example, determining $\mathrm{m}(A)$ for a given A takes time proportional to $|\Theta|$ since we can use the binary representation of A to follow the branches along the binary tree. Another operation which is faster is, for given $A \subseteq \Theta$, determining $\mathrm{Bel}(A)$.

3.1.4 Conversion between the three representations

Each of the three representations has its advantages, so it can be useful to be able to move between different representations.

Converting between the Ordered List and Binary Tree representations can be done very efficiently, in either direction in time proportional to $|\mathcal{F}||\Theta|$.

A natural way of converting a mass potential from Ordered List (or Binary Tree) representation to n-dimensional array representation, is first to initialise the array all to zeros, and then to insert the masses into their correct place. The total time is then related to $2^{|\Theta|}$. However, if it were possible (this depends very much on computer language and implementation used) to bypass the initialisation part, the operation would take time proportional to $|\mathcal{F}||\Theta|$.

The other direction, converting from n-dimensional array representation to Ordered List (or Binary Tree) representation is clearly related to $2^{|\Theta|}$.

3.2 Conditioning a Mass Potential

Let m be a mass potential and B be a subset of Θ. Define m'_B, the *unnormalised conditional of* m *given* B, by, for $A \subseteq \Theta$, $\mathrm{m}'_B(A) = \sum_{C:C\cap B=A} \mathrm{m}(C)$, which, when $A \subseteq B$, is equal to $\sum_{D\subseteq \bar{B}} \mathrm{m}(A \cup D)$.

The mass function m_B, the *conditional of* m *given* B, is defined to be m'_B normalised, and is hence given by: $\mathrm{m}_B(\emptyset) = 0$, and for $A \neq \emptyset$, $\mathrm{m}_B(A) = K^{-1} \sum_{C:C\cap B=A} \mathrm{m}(C)$ where $K = \sum_{C:C\cap B\neq\emptyset} \mathrm{m}(C)$. The conditioned mass function m_B is only well-defined if there exists $C \in \mathcal{F}_\mathrm{m}$ with $C \cap B \neq \emptyset$.

To calculate a conditioned mass function m_B from a mass potential m we can first calculate the unnormalised conditional and then normalise it (the normalisation factor K can be calculated in the first stage). To do this in the Ordered List representation we can first create a new list: we go through the Ordered List, and whenever we find a pair (C, r) with $C \cap B \neq \emptyset$ we add the pair $(C \cap B, r)$ to our new list (where the sets are represented as boolean arrays of size $|B|$). We then sort the list, removing repeats, as described above in section 3.1.2. Finally we normalise. The total number of operations required is proportional to $|\mathcal{F}| \times |B| \times (\log|\mathcal{F}| + \log|\Theta|)$.

The complexity is approximately similar using the Binary Tree representation. Conditioning using the n-dimensional array representation is exponential in $|\Theta|$.

3.3 Calculating the Combined Core of Several Mass Potentials

Many of the algorithms given below have complexity related to the size of the frame Θ. This suggests that if we could reduce the size of the frame, without

losing any information, then the complexity of those algorithms could be considerably improved. This especially applies to some uses of the Fast Möbius transform (see section 4.2), which is exponential in $|\Theta|$.

The *core* $\mathcal{C}_{\mathrm{m}}$ of a mass potential m is the union of its focal sets, that is, $\bigcup_{A\in\mathcal{F}} A$ (see [Shafer, 76], p40). Define the *combined core* of a number of mass potentials $\mathrm{m}_1, \ldots, \mathrm{m}_k$ to be the intersection of their cores, $\mathcal{C}_{\mathrm{m}_1} \cap \cdots \cap \mathcal{C}_{\mathrm{m}_k}$. This is, in fact, the core of their combination $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$ (see section 2.4), if the latter is defined; otherwise it is empty.

All the methods given below for calculating the effect of Dempster's rule can sometimes be very significantly improved by first conditioning all the constituent mass functions by the combined core (this doesn't change the result). As well as reducing the size of the frame, this will sometimes eliminate many of the focal sets of the constituent mass functions.

The combined core can be calculated in time proportional to $|\Theta| \sum_{i=1}^{k} |\mathcal{F}_{\mathrm{m}_i}|$ using the Ordered List or Binary Tree representations. The operation using the n-dimensional Array representation is exponential in $|\Theta|$.

3.4 Lossless Coarsening

Another way of reducing the size of the frame is to merge some elements together (known as coarsening [Shafer, 76]), which in certain cases results in no loss of information.

Define a *coarsening of* Θ to be a pair[7] (Θ', τ) where τ is a function from Θ *onto* Θ' (so that $\tau(\Theta) = \Theta'$). Define the function $\tau^{-1} : 2^{\Theta'} \mapsto 2^{\Theta}$ by $\tau^{-1}(A) = \{\theta \in \Theta : \tau(\theta) \in A\}$. Function $\rho : 2^{\Theta'} \mapsto 2^{\Theta}$ is said to be a *refining* if there exists coarsening (Θ', τ) of Θ with $\rho = \tau^{-1}$ (this can be shown to be equivalent to the definition in [Shafer, 76], p115).

The coarsening (Θ', τ) is essentially characterised by the equivalence relation $\approx_\tau$ defined by $\theta \approx_\tau \psi$ iff $\tau(\theta) = \tau(\psi)$; for if (Θ'', ν) is another coarsening of Θ with $\approx_\nu = \approx_\tau$ then Θ' and Θ'' are the same size, and one can be viewed as just a relabelling of the other.

For any equivalence relation $\approx$ on Θ we can find a coarsening (Θ', τ) with $\approx_\tau = \approx$. For example, we could let Θ' be the set of $\approx$-equivalence classes of Θ, and τ be the associated projection; we call this the *canonical coarsening corresponding to* $\approx$.

3.4.1 Losslessness

Let m be a mass potential over Θ. Equivalence relation $\approx$ on Θ is said to be *lossless for* m if each $A \in \mathcal{F}_{\mathrm{m}}$ is a union of equivalence classes[8] of $\approx$. Coarsening

[7] This differs slightly from Shafer's use of term, [Shafer, 76], p116: he calls Θ' itself a coarsening.

[8] The union of an empty set of sets is taken to be the empty set.

(Θ', τ) is said to be *lossless for* m if $\approx_\tau$ is lossless for m. In this case define the induced mass potential m^τ over Θ' by $\mathrm{m}^\tau(B) = \mathrm{m}(\tau^{-1}(B))$, for $B \subseteq \Theta'$. The sets of focal sets, $\mathcal{F}_\mathrm{m}$ and $\mathcal{F}_{\mathrm{m}^\tau}$ are in 1-1 correspondence and the corresponding values of mass are the same. For $A \subseteq \Theta$, $\mathrm{m}(A) = \mathrm{m}^\tau(B)$ if there exists $B \subseteq \Theta'$ such that $A = \tau^{-1}(B)$, and otherwise $\mathrm{m}(A) = 0$. Hence if we know τ and m^τ, without knowing m, we can recover m. Thus we have expressed the mass potential m more compactly, by using the coarsened frame, *without losing any information*.

In fact m^τ can be viewed just as a shorter way of writing m if Θ' is interpreted as a compact representation of Θ, so that $\theta' \in \Theta'$ is taken to just be an abbreviation of the set $\tau^{-1}(\{\theta'\})$.

Many operations on m can be done more efficiently using m^τ, such as converting a mass function into a belief function, commonality function or plausibility function.

If coarsening (Θ', τ) is lossless for each of $\mathrm{m}_1, \ldots, \mathrm{m}_k$ then it is lossless for $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$. The converse holds, for example, if the core of each of the mass potentials is the same (which will be the case if we have conditioned each mass potential by the combined core, prior to combination): if $\mathcal{C}_{\mathrm{m}_i} = \mathcal{C}_{\mathrm{m}_j}$ for $i, j = 1, \ldots, k$ then (Θ', τ) is lossless for $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$ iff it is lossless for each of $\mathrm{m}_1, \ldots, \mathrm{m}_k$.

Calculation of mass potential induced by lossless coarsening Let (Θ', τ) be a lossless coarsening for mass potential m over Θ. We will assume that m is represented either as an ordered list or as a binary tree. First, let us use the ordering on Θ to define an ordering on Θ'. For each equivalence class E of $\approx_\tau$ let θ^E be its smallest element, and let Θ^* be the set of all θ^E. The function τ restricted to Θ^* (call it τ^*) is a bijection between Θ^* and Θ', and so induces a total ordering on Θ'. Equivalently, we may define the ordering by, for $\varphi, \psi \in \Theta'$, $\varphi < \psi$ iff the smallest element of $\tau^{-1}(\varphi)$ is less than the smallest element of $\tau^{-1}(\psi)$.

Each pair (A, r) (for $A \in \mathcal{F}_\mathrm{m}$) in the representation of m gets mapped to pair $(\tau(A), r)$, where $\tau(A)$ can be efficiently calculated using the equation $\tau(A) = \tau^*(A \cap \Theta^*)$. In the binary tree representation of m_τ these pairs are incrementally added to the binary tree; in the ordered list representation, the pairs will need to be sorted. The computation for the ordered list can be performed in time at worst proportional to $\max\big(|\Theta| \log |\Theta'|,\ |\mathcal{F}_\mathrm{m}|\,|\Theta'|(\log |\mathcal{F}_\mathrm{m}| + \log |\Theta|)\big)$; the computation using the binary tree representation is a little more efficient, being proportional to $\max(|\Theta| \log |\Theta'|,\ |\mathcal{F}_\mathrm{m}|\,|\Theta'| \log |\Theta|)$.

3.4.2 Coarsest lossless coarsening

For mass potential m over Θ define equivalence relation $\approx_\mathrm{m}$ on Θ by $\theta \approx_\mathrm{m} \psi$ if and only if for all $A \in \mathcal{F}_\mathrm{m}$, $\theta \in A \iff \psi \in A$. The equivalence relation $\approx_\mathrm{m}$ can easily be shown to be lossless for m; in fact equivalence relation $\approx$ is

lossless for m if and only if $\approx \subseteq \approx_{\mathrm{m}}$ (i.e., $\theta \approx \psi$ implies $\theta \approx_{\mathrm{m}} \psi$); therefore $\approx_{\mathrm{m}}$ is the unique maximal lossless equivalence relation for m. Hence any coarsening (Θ', τ) with $\approx_\tau = \approx_{\mathrm{m}}$ (such as the canonical coarsening corresponding to $\approx_{\mathrm{m}}$) can be considered a coarsest lossless coarsening since it has minimal Θ'.

Let $\mathcal{M} = \{\mathrm{m}_1, \ldots, \mathrm{m}_k\}$ be a set of mass potentials. Define equivalence relation $\approx_{\mathcal{M}}$ by $\theta \approx_{\mathcal{M}} \psi$ if and only if for all $i = 1, \ldots, k$ and for all $A \in \mathcal{F}_{\mathrm{m}_i}$, $\theta \in A \iff \psi \in A$. Hence $\approx_{\mathcal{M}} = \bigcap_{i=1}^{k} \approx_{\mathrm{m}_i}$, i.e., $\theta \approx_{\mathcal{M}} \psi$ iff for all $i = 1, \ldots, k$, $\theta \approx_{\mathrm{m}_i} \psi$. Clearly $\approx_{\mathcal{M}}$ is the largest equivalence relation which is lossless for $\mathrm{m}_1, \ldots, \mathrm{m}_k$. The canonical coarsening corresponding to $\approx_{\mathcal{M}}$ (call it $(\Theta_{\mathcal{M}}, \tau_{\mathcal{M}})$) is a coarsest lossless coarsening for $\mathrm{m}_1, \ldots, \mathrm{m}_k$.

By the results of the previous subsection, $(\Theta_{\mathcal{M}}, \tau_{\mathcal{M}})$ is lossless for $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$, and if the core of each mass potential is the same, it is the coarsest lossless coarsening for $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$.

Finding coarsest lossless coarsening of a set of mass potentials Again it will be assumed that the mass potentials are represented as either ordered lists or binary trees. To calculate a coarsest lossless coarsening for $\mathcal{M} = \{\mathrm{m}_1, \ldots, \mathrm{m}_k\}$ we determine the list of equivalence classes of $\approx_{\mathcal{M}}$. Throughout the algorithm, structure L is a list of lists and will end up listing the equivalence classes of $\approx_{\mathcal{M}}$. An inner list E of L is a list of integers (in fact numbers $1, \ldots, |\Theta|$) in ascending order, and represents a subset of Θ; for example the subset $\{\theta_4, \theta_7\}$ is represented by the list $[4, 7]$.

First we initialise L to be the one element list containing Θ, that is, $[\,[1, 2, \ldots, |\Theta|]\,]$. We then proceed as follows:

for $i = 1, \ldots, k$
  for $A_i \in \mathcal{F}_{\mathrm{m}_i}$
    for E **in** L
      if both $E \cap A_i$ **and** $E - A_i$ **are non-empty,**
        delete E **from list and insert** $E \cap A_i$ **and** $E - A_i$

The number of operations needed for each E in the inner loop is proportional to $|E| \log |\Theta|$ so, since $\sum_{E \in L} |E| = |\Theta|$, the total number operations for the algorithm is proportional to $|\Theta|(\log |\Theta|) \sum_{i=1}^{k} |\mathcal{F}_{\mathrm{m}_i}|$.

Hence calculating the coarsest lossless coarsening Θ' of a number of mass potentials $\mathrm{m}_1 \ldots, \mathrm{m}_k$ over Θ, and converting each to a mass potential over Θ' can be done in time proportional to $|\Theta|(\log |\Theta|) \sum_{i=1}^{k} |\mathcal{F}_{\mathrm{m}_i}|$.

4 Conversion Between m, Bel and Q

Mass function m, its associated belief function Bel, and its associated commonality function Q all, in a sense, contain the same information, since it is possible to reconstruct the other two from any one of the three. As mentioned above,

mass functions/potentials are often the most compact way of representing the information, as often there are only a relatively small number of focal sets. However some operations are more convenient with one of the other functions; Dempster's rule has a very simple form for commonality functions (see 2.4), so it can sometimes be useful to convert to commonality functions before combining; it is usually values of belief that in the end we're interested in, and if we want a large number of values of belief it can sometimes be easier to convert to the belief function representation to read these off.

Therefore it is important to be able to move efficiently between the three representations. This can be done with what is known as the *Fast Möbius Transformation* (FMT) [Thoma, 89; 91; Kennes and Smets, 90a,b].

4.1 Relationships between the Various Functions

In section 2.1 we defined the belief function and the commonality function associated with a mass function; we can generalise these definitions to mass potentials.

Let m be a mass potential over Θ. The associated *unnormalised belief function* $\mathrm{Bel_m}$ is defined by, for $A \subseteq \Theta$, $\mathrm{Bel_m}(A) = \sum_{B \subseteq A} \mathrm{m}(B)$. The associated *unnormalised commonality function* $\mathrm{Q_m}$ is defined by, for $A \subseteq \Theta$, $\mathrm{Q_m}(A) = \sum_{B \supseteq A} \mathrm{m}(B)$.

$\mathrm{Bel_m}$ and $\mathrm{Q_m}$ will often be abbreviated to Bel and Q respectively, when it is clear from the context which is the associated mass potential.

The mass potential m can be recovered from its associated unnormalised belief function Bel by using the equation

$$\text{For } A \subseteq \Theta, \quad \mathrm{m}(A) = \sum_{B \subseteq A} (-1)^{|A-B|} \mathrm{Bel}(B),$$

and, similarly, m can be recovered from its associated unnormalised commonality function Q with the equation

$$\text{For } A \subseteq \Theta, \quad \mathrm{m}(A) = \sum_{B \supseteq A} (-1)^{|B-A|} \mathrm{Q}(B).$$

These results follow easily from lemma 2.3, page 48 of [Shafer, 76]. We can also use exactly the same proof as that for Theorem 2.4 in [Shafer, 76] to give the direct relationship between unnormalised belief function Bel and unnormalised commonality function Q which correspond to the same mass potential:

$$\mathrm{Bel}(A) = \sum_{B \subseteq \bar{A}} (-1)^{|B|} \mathrm{Q}(B)$$

and

$$\mathrm{Q}(A) = \sum_{B \subseteq A} (-1)^{|B|} \mathrm{Bel}(\bar{B})$$

for all $A \subseteq \Theta$.

4.2 The Fast Möbius Transform

As usual we enumerate Θ as $\{\theta_1, \ldots, \theta_n\}$, and we'll view subset A of Θ as an n-digit binary number `A` where the most significant digit is 1 iff $\theta_1 \in A$. We will use an n-dimensional array $\{0,1\} \times \cdots \times \{0,1\}$ of reals in $[\,0,1\,]$ which we shall call v (see section 3.1.1) to represent each of the functions m, Bel and Q. For example, if we are representing Q then the value of $Q(A)$ will be placed in position `A` of the array.

4.2.1 The conversion algorithms

Suppose we want to calculate unnormalised belief function Bel from mass potential m. The most obvious algorithm involves, for each $A \subseteq \Theta$, calculating $\mathrm{Bel}(A)$ by looking up $\mathrm{m}(B)$ for each $B \subseteq A$, and summing them. The number of additions required is $\sum_{A \subseteq \Theta} 2^{|A|} = \sum_{j=1}^{n} \binom{n}{j} 2^j = (2+1)^n = 3^n$. However this algorithm is very wasteful, as the same value of $\mathrm{m}(B)$ is looked up many times.

Instead we will initialise the array v to mass potential m and pass the mass $\mathrm{m}(A)$ up to all supersets of A.

(i) Algorithm for converting for all $A \subseteq \Theta$ $v(A)$ to $\sum_{B \subseteq A} v(B)$

for $\theta \in \Theta$
  for $A \subseteq \Theta - \{\theta\}$
    $v(A \cup \{\theta\})$ **:=** $v(A \cup \{\theta\}) + v(A)$

The outer loop could be implemented as a loop '**for** $j = 1$ **to** n', with θ being labelled θ_j. The inner loop could be implemented using nested two-valued loops, each loop corresponding to an element θ_i of Θ and determining whether θ_i is in A or not (extra control statements are also required).

Let $v_I(A)$ be the initial value of $v(A)$ (the input) and let $v_O(A)$ be the final value of $v(A)$ (the output). For the algorithm to be correct we need that, for all $A \subseteq \Theta$, $v_O(A) = \sum_{B \subseteq A} v_I(B)$. Now, for each $A \subseteq \Theta$, $v_O(A)$ can be written as a sum of terms $v_I(B)$ for various B (this can be proved using an obvious induction); moreover it can be seen that $v_I(B)$ appears in this summation for $v_O(A)$ iff $B \subseteq A$; finally it can be checked that each term $v_I(B)$ can appear at most once in the term $v_O(A)$ — this is because there is a unique path that each $v_I(B)$ follows to reach $v_O(A)$, for $B \subseteq A$. Hence $v_O(A) = \sum_{B \subseteq A} v_I(B)$.

The correctness of very closely related algorithms (ii), (iii) and (iv) below follows by similar arguments.

To calculate the function Bel from m we initialise the array v to m, and apply algorithm (i) above: the array v will then be set to Bel.

(ii) Algorithm[9] **for converting for all** $A \subseteq \Theta$ $v(A)$ **to** $\sum_{B \supseteq A} v(B)$

for $\theta \in \Theta$
  for $A \subseteq \Theta - \{\theta\}$
    $v(A) := v(A) + v(A \cup \{\theta\})$

To calculate the function Q from m we initialise the array v to m, and apply algorithm (ii) above: the array v will then be set to Q.

(iii) Algorithm for converting for all $A \subseteq \Theta$ $v(A)$ **to** $\sum_{B \subseteq A} (-1)^{|A-B|} v(B)$

for $\theta \in \Theta$
  for $A \subseteq \Theta - \{\theta\}$
    $v(A \cup \{\theta\}) := v(A \cup \{\theta\}) - v(A)$

This can be used to calculate m from Bel: if we initialise the array v to Bel and apply this algorithm, the final state of the array gives m, i.e., if v_O is the final state of the array, we will have, for all $A \subseteq \Theta$, $v_O(A) = \mathrm{m}(A)$.

Algorithm (iii) can also be used to convert between Bel and Q. If we initialise the array v to Q, and apply the algorithm we can recover Bel from the output v_O by using the equation, for $A \subseteq \Theta$, $\mathrm{Bel}(A) = |v_O(\bar{A})|$. This is because $\mathrm{Bel}(\bar{A}) = \sum_{B \subseteq A} (-1)^{|B|} \mathrm{Q}(B) = |\sum_{B \subseteq A} (-1)^{|A-B|} \mathrm{Q}(B)|$.

If, on the other hand, we want to calculate the unnormalised commonality function Q from the unnormalised belief function Bel we initialise array v by setting $v(A) = \mathrm{Bel}(\bar{A})$, apply algorithm (iii) to get output v_O. For $A \subseteq \Theta$, $\mathrm{Q}(A) = |v_O(A)|$.

(iv) Algorithm for converting for all $A \subseteq \Theta$ $v(A)$ **to** $\sum_{B \supseteq A} (-1)^{|B-A|} v(B)$

for $\theta \in \Theta$
  for $A \subseteq \Theta - \{\theta\}$
    $v(A) := v(A) - v(A \cup \{\theta\})$

To calculate m from Q we initialise v with Q and apply the algorithm to give m.

4.2.2 The time needed for conversion

The number of additions used by each of the algorithms is $n2^{n-1}$ where $n = |\Theta|$, and the number of other basic operations needed for the algorithm (such as incrementing of loop counters) is of similar order, so one might use this figure

[9] This is clearly very strongly related to algorithm (i); in fact one might even consider it to be actually the *same* algorithm, if, in the binary representation of a set, a 1 is reinterpreted as meaning that the element is *not* in the set (rather than that it is in the set).

as a measure of the complexity. However, to be consistent with the assumptions in section 3 (see 3.1.2) we need to say that accessing a value of v takes time proportional to n, so overall the time needed is proportional to $n^2 2^n$.

4.2.3 Use of conditioning by core and lossless coarsening

Clearly reducing the size of $|\Theta|$ can hugely improve the efficiency of the above algorithms. For example if we manage to reduce it by 10 elements then it makes the algorithm more than 1000 times faster. Conditioning by the combined core (sections 3.2 and 3.3) could be used when we know we are later going to combine with a set of other mass potentials; lossless coarsening (section 3.4) could be used as long as the number of focal sets is not huge. For either of these methods to be efficient, we need that the list of focal sets is also stored in a more direct way than the n-dimensional array, e.g., using the ordered list or binary tree representations.

5 Exact Combination on Frames

As mentioned earlier, the central computational problem of Dempster-Shafer theory is calculating the combination of a number of mass functions. It is assumed that the input is a number of mass functions $\mathrm{m}_1, \ldots, \mathrm{m}_k$, and we are interested in the combination $\mathrm{m} = \mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$, especially the associated values of belief (values of Bel_m) for various sets of interest.

Here we look at some exact methods. Further methods are described in sections 6,7, 9 and 10.

Section 5.1 considers an obvious approach to the problem; section 5.2 describes the use of the Fast Möbius Transform to compute the combination. Section 5.3 takes a different approach, computing directly a value of combined belief, without first computing the combined mass function.

We'll actually consider the slightly more general problem of calculating the combination of a number of mass potentials. Recall a mass potential is defined to be a function from 2^Θ to $[\,0, 1\,]$, and a mass function is a mass potential m such that $\mathrm{m}(\emptyset) = 0$ and $\sum_{A \in 2^\Theta} \mathrm{m}(A) = 1$.

The unnormalised combination $\mathrm{m}_1 \otimes \mathrm{m}_2$ of mass potentials m_1 and m_2 over Θ is defined by, for $A \subseteq \Theta$,

$$(\mathrm{m}_1 \otimes \mathrm{m}_2)(A) \quad = \sum_{B,C\,:\,B \cap C = A} \mathrm{m}_1(B)\mathrm{m}_2(C).$$

Define the combination $\mathrm{m}_1 \oplus \mathrm{m}_2$ of m_1 and m_2 to be the normalisation of $\mathrm{m}_1 \otimes \mathrm{m}_2$ (when this is proper): $(\mathrm{m}_1 \oplus \mathrm{m}_2)(\emptyset) = 0$, and for $A \neq \emptyset$, $(\mathrm{m}_1 \oplus \mathrm{m}_2)(A) = K_{1,2}\,(\mathrm{m}_1 \otimes \mathrm{m}_2)(A)$ where the normalisation constant $K_{1,2}$ is given by $K_{1,2}^{-1} = \sum_{A \neq \emptyset} (\mathrm{m}_1 \otimes \mathrm{m}_2)(A)$ which equals $\sum_{B \cap C \neq \emptyset} \mathrm{m}_1(B)\mathrm{m}_2(C)$.

$\mathrm{m}_1 \oplus \mathrm{m}_2$ is a mass function, and when m_1 and m_2 are mass functions this definition agrees with the standard definition of Dempster's rule (given in section 2.4).

Both $\otimes$ and $\oplus$ are commutative and associative. For mass potentials $\mathrm{m}_1, \ldots, \mathrm{m}_k$, their unnormalised combination $\bigotimes_{i=1}^{k} \mathrm{m}_i$ can be shown to be given by

$$\bigotimes_{i=1}^{k} \mathrm{m}_i(A) = \sum_{B_1 \ldots, B_k \,:\, B_1 \cap \cdots \cap B_k = A} \mathrm{m}_1(B_1) \cdots \mathrm{m}_k(B_k).$$

As one would hope, $\bigoplus_{i=1}^{k} \mathrm{m}_i$ is the normalisation of $\bigotimes_{i=1}^{k} \mathrm{m}_i$, which means that when we're calculating $\bigoplus_{i=1}^{k} \mathrm{m}_i$ we can always work with unnormalised combination, and leave normalisation until the end. Also, the unnormalised commonality function of $\bigotimes_{i=1}^{k} \mathrm{m}_i$ is $\prod_{i=1}^{k} \mathrm{Q}_{\mathrm{m}_i}$.

5.1 Combination Using Mass Potentials

We suppose that we are given mass potentials $\mathrm{m}_1, \ldots, \mathrm{m}_k$ over Θ, which are represented as ordered lists or binary trees (see section 3.1), and we wish to calculate their combination $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$. This combined mass function produced can then be used, for example, to calculate the values of combined belief for sets of interest.

The method described in section 5.1 is a refinement of the obvious algorithm sketched in [Shafer, 76], chapter 3.

5.1.1 Combination of two mass potentials

Suppose we wish to calculate $\mathrm{m}_1 \oplus \mathrm{m}_2$ for mass potentials m_1 and m_2 over Θ. We will proceed by calculating the unnormalised combination $\mathrm{m}_1 \otimes \mathrm{m}_2$, and then normalise.

To calculate $\mathrm{m}_1 \otimes \mathrm{m}_2$ we can use algorithms of the following form (recall $\mathcal{F}_{\mathrm{m}}$ is the set of focal sets of mass potential m): First initialise the appropriate data structure. Then, for each $A \in \mathcal{F}_{\mathrm{m}_1}$ and $B \in \mathcal{F}_{\mathrm{m}_2}$, compute $A \cap B$ and $\mathrm{m}(A)\mathrm{m}(B)$ and add the pair $\big(A \cap B, \mathrm{m}(A)\mathrm{m}(B)\big)$ to the data structure for $\mathrm{m}_1 \otimes \mathrm{m}_2$.

The initialisation step and the definition of "add" here depends on the data structure we use to store the combined mass potential.

If we use a binary tree for $\mathrm{m}_1 \otimes \mathrm{m}_2$ then we incrementally build the binary tree from nothing; adding $\big(A \cap B, \mathrm{m}(A)\mathrm{m}(B)\big)$ to the tree means here first finding if there is a leaf $(A \cap B, r)$ for some r; if there is then we replace r by $r + \mathrm{m}(A)\mathrm{m}(B)$; if there is no such leaf, we insert a leaf $\big(A \cap B, \mathrm{m}(A)\mathrm{m}(B)\big)$ along with the appropriate extra internal node. Computing the combination $\mathrm{m}_1 \oplus \mathrm{m}_2$ hence can be performed in time proportional to $|\mathcal{F}_{\mathrm{m}_1}|\,|\mathcal{F}_{\mathrm{m}_2}|\,|\Theta|$.

If, on the other hand, we want to use an ordered list to represent $\mathrm{m}_1 \otimes \mathrm{m}_2$, we can incrementally build up an ordered list from an initial empty list, adding

the pair $\big(A \cap B, \mathrm{m}(A)\mathrm{m}(B)\big)$ to the list in the correct position, in a similar way as for the binary tree. The disadvantage of this method is that finding the correct place to add the pair is a little more time-consuming than for a binary tree. An alternative is to incrementally build up an unordered list, adding $\big(A \cap B, \mathrm{m}(A)\mathrm{m}(B)\big)$ to the end of the list, and then afterwards sort the list. This has roughly similar complexity to the algorithm with a binary tree, the disadvantage being that it uses more space, proportional to $|\mathcal{F}_{\mathrm{m}_1}|\,|\mathcal{F}_{\mathrm{m}_2}|\,|\Theta|$.

Because of these disadvantages of the ordered list representation of $\mathrm{m}_1 \otimes \mathrm{m}_2$ it may well often be better to use the binary tree representation, and then, if we want the output to be stored as an ordered list, to convert afterwards to the ordered list representation.

If the number of focal sets, for each of m_1 and m_2, is close to the maximum value of $2^{|\Theta|}$ then the above methods need time roughly proportional to $|\Theta|2^{2|\Theta|}$. This is much worse that the FMT method described in section 5.2. However, if both mass potentials only have a small number of focal sets, this simple approach will be much faster than the FMT method.

5.1.2 Combination of several mass potentials

Suppose we are given a set of mass potentials over Θ, and we wish to calculate their combination; we assume that they are represented as ordered lists or binary trees. Before proceeding further it's a good idea to find the combined core and condition all the mass potentials by it (see sections 3.2 and 3.3), as this can reduce the number of focal sets. We can also find the coarsest common refinement Θ', and coarsen to this frame (see 3.4), but this is of limited use here, as it doesn't reduce the number of focal sets; however it does tell us that the number of possible focal sets of the combined mass potential is at most $2^{|\Theta'|}$, which, if Θ' is fairly small, can give us useful information about how long the calculation will take.

The next step is to order the resulting mass potentials as $\mathrm{m}_1, \ldots, \mathrm{m}_k$. As we shall see, the choice of ordering can make a huge difference. We use the algorithm (section 5.1.1) for combining two mass potentials $k-1$ times to calculate first $\mathrm{m}_1 \otimes \mathrm{m}_2$, and then $(\mathrm{m}_1 \otimes \mathrm{m}_2) \otimes \mathrm{m}_3$, and so on, until we have calculated $\mathrm{m}_1 \otimes \cdots \otimes \mathrm{m}_k$ as $(\mathrm{m}_1 \otimes \cdots \otimes \mathrm{m}_{k-1}) \otimes \mathrm{m}_k$. Finally we normalise to produce $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$.

For $i = 1, \ldots, k$, let $\mathcal{F}_i = \mathcal{F}_{\mathrm{m}_i}$, the set of focal sets of m_i, and let $\mathcal{F}_{1,\ldots,i}$ be the set of focal sets of $\mathrm{m}_1 \otimes \cdots \otimes \mathrm{m}_i$; abbreviate the set of focal sets of the whole unnormalised combination, $\mathcal{F}_{1,\ldots,k}$ to just $\mathcal{F}$. The time needed for the computation is proportional to $R = |\Theta| \sum_{i=1}^{k-1} |\mathcal{F}_{1,\ldots,i}|\,|\mathcal{F}_{i+1}|$.

An upper bound for $|\mathcal{F}_{1,\ldots,i}|$ is $\prod_{j=1}^{i} |\mathcal{F}_j|$, so $R \le |\Theta| \sum_{i=1}^{k-1} \prod_{j=1}^{i+1} |\mathcal{F}_j|$ which equals $|\Theta|(1 + \frac{U}{|F_k|}) \prod_{j=1}^{k} |\mathcal{F}_j|$, where $U = 1 + \frac{1}{|F_{k-1}|} + \frac{1}{|F_{k-1}| \times |F_{k-2}|} + \cdots + \prod_{i=3}^{k-1} \frac{1}{|F_i|}$. Assuming that each m_i has at least two focal sets (which we can ensure e.g., by first combining all the mass potentials with only one focal set,

by taking the intersection of these sets, and then conditioning one of the other mass potentials by this intersection), $U \leq 2$ so an upper bound for R is $|\Theta|(1 + \frac{2}{|\mathcal{F}_k|}) \prod_{i=1}^{k} |\mathcal{F}_i|$ which is at most $2|\Theta| \prod_{i=1}^{k} |\mathcal{F}_i|$.

Example Let $\Theta = \{\theta_1, \ldots, \theta_n\}$, let $k = n$ and for $i = 1, \ldots, k$, define mass function m_i by $\mathrm{m}_i(\Theta) = 0.5$ and $\mathrm{m}_i(\Theta - \{\theta_i\}) = 0.5$. In this case we indeed have $|\mathcal{F}_{1,\ldots,i}| = \prod_{j=1}^{i} |\mathcal{F}_j| = 2^i$, and $|\mathcal{F}|$ has the maximum possible value $2^n = 2^k$. Here R can be seen to be $n(2^{k+1} - 4)$.

From this example it can be seen that this method of computing the combination can be as bad as exponential in $\min(|\Theta|, k)$.

However, the last example is rather an extreme case, and usually the number of focal sets of the whole combination $|\mathcal{F}|$ will be much less than its maximum possible value $\prod_{i=1}^{k} |\mathcal{F}_i|$ (although it may well be still typically exponential in k). Usually a better upper bound for R is

$$|\Theta| \left(\max_{i=1}^{k-1} |\mathcal{F}_{1,\ldots,i}| \right) \sum_{i=2}^{k} |\mathcal{F}_i|.$$

We would tend to expect $|\mathcal{F}_{1,\ldots,i}|$ to be an increasing function of i (usually sharply increasing), especially if before combination we conditioned all the mass potentials by their combined core, since more combinations will tend to produce more focal sets. It is always monotonically increasing if, for example, Θ is in each $\mathcal{F}_i$. (However, as we will shortly see, it is not always the case that $|\mathcal{F}_{1,\ldots,i}|$ is an increasing function of i, and it is not even always possible to reorder the mass potentials to make it increasing in i.)

If $|\mathcal{F}_{1,\ldots,i}|$ is an increasing function of i then an upper bound for R is $|\Theta| \, |\mathcal{F}| \sum_{i=2}^{k} |\mathcal{F}_i|$. If $|\mathcal{F}_{1,\ldots,i}|$ is a sharply increasing function of i, and none of the sets $\mathcal{F}_i$ are too large, we would expect the last term (i.e., that corresponding to the last combination) to be dominant, which is less than $|\Theta| \, |\mathcal{F}| \, |\mathcal{F}_k|$.

This computation of the combination clearly takes time at least proportional to $|\Theta| \, |\mathcal{F}|$; often when the sets $\mathcal{F}_i$ are not large, the computation time will be fairly closely tied to this term $|\Theta| \, |\mathcal{F}|$.

The order of the mass potentials That the order of the mass potentials can make a huge difference is illustrated by adding to the previous example another (Bayesian) mass potential whose focal sets are all the singletons $\{\theta_j\}$ for $\theta_j \in \Theta$.

If this mass potential is added to the end of the list of mass potentials then the computation is exponential in n; if it is added to the beginning of the list, the computation is cubic in n $(= k - 1)$ (and could even be made quadratic) since the number of focal sets, $|\mathcal{F}_{1,\ldots,i}|$, never exceeds $n + 1$.

This example also shows that $|\mathcal{F}_{1,\ldots,i}|$ is not always increasing in i, since $|\mathcal{F}| = |\mathcal{F}_{1,\ldots,k}| = n+1$, but when the Bayesian mass potential is added at the end, $|\mathcal{F}_{1,\ldots,k-1}|$ is 2^n.

A natural question to ask is whether we can always reorder the mass potentials to ensure that $|\mathcal{F}_{1,\ldots,i}|$ is monotonically increasing in i. The following example emphatically shows that we cannot always do so:

For $i = 1, \ldots, n$, let m_i be such that $\mathcal{F}_i = \{\{\theta_i\}\} \cup 2^{\Theta - \{\theta_i\}}$. Here $|\mathcal{F}_i| = 2^{n-1} + 1$ and $\mathcal{F}$ just consists of the empty set and the singletons, so $|\mathcal{F}| = n+1$. $|\mathcal{F}_{1,\ldots,i}| = 2^{n-i} + i$ so it is exponentially *decreasing* in i. The example is symmetrical so changing the order of the mass potentials makes no difference.

A heuristic method for deciding order of mass potentials Since the order in which we combine a set of mass potentials can be so important, this raises the question of how we choose to order them. One fairly simple heuristic method which deals with the problem of the Bayesian mass function in the penultimate example is to calculate, for each mass potential m, the average size of its focal sets, that is $\frac{1}{|\mathcal{F}_{\mathrm{m}}|} \sum_{A \in \mathcal{F}_{\mathrm{m}}} |A|$; we order the mass potentials as $\mathrm{m}_1, \ldots, \mathrm{m}_k$, with for $i < j$ the average size of focal sets for m_i being not more than that for m_j. This method is easy and efficient to implement; however, it is not clear how well it does generally at finding an order close to the optimal one.

Direct computation An alternative to this incremental method is to generalise the algorithm given for combining two mass potentials, labelling the mass potentials $\mathrm{m}_1, \ldots, \mathrm{m}_k$:

For each $i = 1, \ldots k$ and $A_i \in \mathcal{F}_{\mathrm{m}_i}$, add pair $(A_1 \cap \cdots \cap A_k, \mathrm{m}_1(A_1) \cdots \mathrm{m}_k(A_k))$ to the data structure for the combination.

However this certainly does not appear to be usually the best way of performing the combination: the number of operations needed is at least proportional to $|\Theta| \prod_{i=1}^{k} |\mathcal{F}_{\mathrm{m}_i}|$ The efficiency can be seen to be, at best, similar to the incremental algorithm given above, and, at worst, exponentially worse.

5.2 Calculation of Dempster's Rule using Fast Möbius Transform

The Fast Möbius Transform can be used to calculate the combination of a number of mass potentials over Θ.

As with previous use of the Fast Möbius Transform, reducing the size of the frame by (i) conditioning by the combined core of the mass functions (see 3.3) and (ii) coarsest lossless coarsening (see 3.4) will in some cases hugely improve the efficiency of the calculation, if we have a representation for the focal sets of each mass potential (e.g., in an ordered list or binary tree), and if the number

of focal sets of each mass potential isn't too large. Note that lossless coarsening can be used in two ways: firstly we can find the coarsest lossless coarsening for each individual mass potential, which makes the conversion to commonalities (see below) more efficient; secondly, we can find the coarsest lossless coarsening for the set of mass potentials, which can make other steps faster as well.

We use Fast Möbius Transform algorithm (ii) on each mass potential m_i, $i = 1, \ldots, k$, to convert each to its associated unnormalised commonality function Q_i. We then calculate unnormalised commonality function Q by pointwise multiplication from the Q_is: for all $A \subseteq \Theta$ let $\mathrm{Q}(A) = \prod_{i=1}^{k} \mathrm{Q}_i(A)$. We then can calculate the mass potential m associated with Q by using Fast Möbius Transform algorithm (iv). Finally we can normalise m to give the combination $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$.

The first stage requires $kn2^{n-1}$ additions; the pointwise multiplication stage requires $(k-1)2^n$ multiplications; the calculation of m from Q requires $n2^{n-1}$ additions, and the normalisation uses one division and $2^n - 1$ multiplications. Hence the number of operations is of order $kn2^n$. We're also assuming (see 3.1.2) that it takes time proportional to n to access an element of one of the n-dimensional arrays, so overall, the algorithm takes time proportional to $kn^2 2^n$.

If we wanted instead of the combined mass function, the belief function associated with $\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$, we could have converted Q directly to the unnormalised belief function, using e.g., Fast Möbius Transform algorithm (iii), and then normalised. This doesn't change the number of operations required. Also we could instead have as inputs belief functions $\mathrm{Bel}_1, \ldots, \mathrm{Bel}_k$ and converted these directly to $\mathrm{Q}_1, \ldots, \mathrm{Q}_k$.

Mass-based or Fast Möbius Transform?

A natural question to ask is, for a given problem, how do we decide which algorithm to use. Sometimes it's clear: if we need the values of combined belief for all 2^n subsets of Θ then, the FMT is better; likewise if one of the mass potentials has a huge number of focal sets (close to 2^n). A nice feature of the Fast Möbius Transform algorithm is that one can accurately estimate how long it will take, in terms of the size of the frame n and number of mass potentials k. With the mass-based algorithm, we just have crude upper bounds. Clearly if these are better than the predicted time for the FMT then we should use the mass-based algorithm. The complexity of the mass-based algorithm is strongly tied to the number of focal sets of the combined belief function; if it can be shown that this is much smaller than 2^n then again, the mass-based method is likely to be much faster.

However, generally we will probably not be able to tell beforehand which of the two is better. For this case, one very simple idea is first to calculate approximately how long the Fast Möbius Transform algorithm will take; then we spend that long attempting to use the mass-based method; if in that time

we don't finish, we use the Fast Möbius Transform algorithm. This, to some extent, combines the advantages of both: it takes at most twice as long as the Fast Möbius Transform, but when the mass-based method is the faster method, it behaves similarly to the latter.

5.3 Direct Calculation of Belief

The idea behind this approach is to calculate a value of combined belief directly without first calculating the combined mass function.

For this method it is convenient to use source triples (see sections 2.2 and 2.4.2) to represent the mass functions. The combined belief function Bel, from source triples $(\Omega_i, \mathrm{P}_i, \Gamma_i)$, for $i = 1, \ldots, k$ is given by: for $A \subseteq \Theta$, $\mathrm{Bel}(A) = \mathrm{P}_{\mathrm{DS}}(\{\omega \in \Omega : \Gamma(\omega) \subseteq A\})$. It then follows that

$$\mathrm{Bel}(A) = \frac{\mathrm{P}'(\Gamma'(\omega) \subseteq A) - \mathrm{P}'(\Gamma'(\omega) = \emptyset)}{1 - \mathrm{P}'(\Gamma'(\omega) = \emptyset)},$$

where $\mathrm{P}'(\Gamma'(\omega) \subseteq A)$ is shorthand for $\mathrm{P}'(\{\omega \in \Omega^{\times} : \Gamma'(\omega) \subseteq A\})$, and recall that $\Gamma'(\omega) = \bigcap_{i=1}^{k} \Gamma_i(\omega(i))$ and $\mathrm{P}'(\omega) = \prod_{i=1}^{k} \mathrm{P}_i(\omega(i))$, for $\omega \in \Omega^{\times}$. Hence the problem of calculating values of Dempster-Shafer belief can be reduced to calculating $\mathrm{P}'(\Gamma'(\omega) \subseteq A)$ for various $A \subseteq \Theta$, since $\mathrm{P}'(\Gamma'(\omega) = \emptyset)$ is given by setting $A = \emptyset$.

We can think of the product set $\Omega^{\times} = \Omega_1 \times \cdots \times \Omega_k$ geometrically as a k-dimensional cube of dimension 1, which is split into hyper-rectangles ω, where ω is of dimensions $\mathrm{P}_1(\omega(1)) \times \cdots \times \mathrm{P}_k(\omega(k))$, and so has (hyper-)volume $\mathrm{P}'(\omega)$. Calculating $\mathrm{P}'(\Gamma'(\omega) \subseteq A)$ then amounts to finding the volume of the region $R = \{\omega : \Gamma'(\omega) \subseteq A\}$. We find this volume by recursively splitting the region into disjoint parts, finding the volumes of these regions, and summing them.

This method is only useful if the number of focal sets of each input mass function, or $|\Omega_i|$, is small (e.g. of order $|\Theta|$ rather than of order $2^{|\Theta|}$).

The approach described here is a simple generalisation of the method described in section 4.1 of [Wilson, 89] for the combination of simple support functions (see also [Wilson, 92c], which includes as well implementation details and experimental results).

There are strong connections between this approach and ATMS-style methods for calculating combined belief e.g., [Laskey and Lehner, 89; Provan, 90; Kohlas and Monney, 95] and chapter 6 in this volume, on probabilistic argumentation systems; however the view we describe here is perhaps more direct, and with the exception of the latter, more general, as it allows arbitrary input mass functions. So, for the sake of concision, this chapter does not include a description of these ATMS-style approaches.

We will continue with this geometric view. In section 5.3.2 we briefly indicate how the ideas can also be thought of in a simple algebraic way.

5.3.1 Recursive splitting up of regions in $\Omega^\times$

Region R can be seen to have a particular structure. $\Gamma'(\omega) \subseteq A$ if and only if for all $\theta \in \bar{A}$, $\Gamma'(\omega) \not\supseteq \theta$, which holds if and only if for all $\theta \in \bar{A}$ there exists $i \in \{1, \ldots, k\}$ such that $\Gamma_i(\omega(i)) \not\supseteq \theta$. Therefore

$$R = \bigcap_{\theta \in \bar{A}} \bigcup_{i=1}^{k} R^i_\theta,$$

where $R^i_\theta = \{\omega \in \Omega^\times : \Gamma_i(\omega(i)) \not\supseteq \theta\}$. Region R^i_θ is a one-dimensional slice of $\Omega^\times$. It can be written as

$$\Omega_1 \times \cdots \times \Omega_{i-1} \times \{\omega_i \in \Omega_i : \Gamma_i(\omega(i)) \not\supseteq \theta\} \times \Omega_{i+1} \times \cdots \times \Omega_k,$$

and so can be thought of as just a subset of Ω_i. The volume of R^i_θ, i.e., $\mathrm{P}'(R^i_\theta)$, is $\mathrm{P}_i(\{\omega_i \in \Omega_i : \Gamma_i(\omega_i) \not\supseteq \theta\})$ which equals $\mathrm{Bel}_i(\overline{\{\theta\}})$, where Bel_i is the belief function associated with the ith source triple.

Calculating Volume of Simple Expressions and Simplifying

The method we describe is based on recursively breaking the expression[10] $\bigcap_{\theta \in \bar{A}} \bigcup_{i=1}^{k} R^i_\theta$ into expressions of a similar form until they are simple enough for their volume to be easily calculated. Before giving the recursive step, we list a number of ways of directly calculating the volume of simple expressions of this form, and ways of simplifying expressions.

Write R as $\bigcap_{\theta \in B} \bigcup_{i \in \sigma_\theta} R^i_\theta$ where B is set to $\bar{A}$ and for each $\theta \in B$, σ_θ is set to $\{1, \ldots, k\}$; however, as the algorithm progresses, it will generate expressions of this form with smaller B and σ_θ. Also write $|R|$ for the volume of R.

(a) *If, for some θ, $\sigma_\theta = \emptyset$ then $R = \emptyset$* so its volume $|R| = 0$. Similarly, if for some θ, each $R^i_\theta = \emptyset$ (for each $i \in \sigma_\theta$) then $|R| = 0$.

(b) *If $B = \emptyset$ then $|R| = 1$.*

(c) *If $|B| = 1$ then $|R|$* can be calculated easily, writing B as $\{\theta\}$, using $1 - |R| = \prod_{i \in \sigma_\theta} (1 - |R^i_\theta|) = \prod_{i \in \sigma_\theta} \mathrm{Pl}_i(\{\theta\})$ where Pl_i is the plausibility function associated with the ith source triple.

(d) *If for some θ and i, $R^i_\theta = \emptyset$ then* we can omit that i from σ_θ.

(e) *If for some θ and $i \in \sigma_\theta$, $R^i_\theta = \Omega^\times$ then $\bigcup_{j \in \sigma_\theta} R^j_\theta = \Omega^\times$* so θ can be omitted from B without changing (the set represented by) R.

(f) *Suppose, for some $\theta, \tau \in B$, that $\sigma_\theta \subseteq \sigma_\tau$ and for each $i \in \sigma_\theta$, $R^i_\theta \subseteq R^i_\tau$. Then τ* can be omitted from B without changing R.

[10] Note that although each R^i_θ and each expression represents a set, for the purpose of the algorithm we are regarding the expressions as formal expressions where each R^i_θ is treated as an atomic symbol. However, the soundness of the rewriting rules such as (a)–(f) below is checked, of course, by reference to the subsets that they refer to.

Splitting

This is the basic recursive step in the method. Before applying this step we first check to see if $|R|$ can be calculated immediately with (a)–(c) or R simplified with (d)–(f). If R can be simplified we do so, and check (a)–(f) again. If none of these can be applied we check to see if we can *factor* (see below); if not we *split*.

To split $R = \bigcap_{\theta \in B} \bigcup_{i \in \sigma_\theta} R^i_\theta$ involves first choosing an l in some σ_θ and then *splitting by* l. Splitting by l involves slicing R into $|\Omega_l|$ disjoint regions $\{R_{\omega_l} : \omega_l \in \Omega_l\}$ where R_{ω_l} is defined to be $\{\omega \in R : \omega(l) = \omega_l\}$. Each R_{ω_l} can be written as $R'_{\omega_l} \times \{\omega_l\}$ where

$$R'_{\omega_l} = \bigcap_{\theta \in B \cap \Gamma_l(\omega_l)} \quad \bigcup_{i \in \sigma_\theta - \{l\}} R^i_\theta,$$

which is of the same form as R, but a smaller expression, so we can recursively repeat the process to calculate $|R'_{\omega_l}|$. Finally $|R|$ can be calculated as

$$\sum_{\omega_l \in \Omega_l} \mathrm{P}_l(\omega_l)\, |R'_{\omega_l}|.$$

Note that if $B \cap \Gamma_l(\omega_l)$ is much smaller than B then the expression for R'_{ω_l} is very much simpler than that for R. Therefore one natural heuristic for choosing which l to split by, is to choose l such that $\sum_{\omega_l \in \Omega_l} |B \cap \Gamma_l(\omega_l)|$ is minimal.

Factoring

This is another recursive step; it simplifies the expression much more than splitting typically does, but can only sometimes be applied.

Suppose there exists non-empty proper subset C of B such that $\bigcup_{\theta \in C} \sigma_\theta \cap \bigcup_{\theta \in B-C} \sigma_\theta = \emptyset$. We then write R as $R_1 \cap R_2$ where factor $R_1 = \bigcap_{\theta \in C} \bigcup_{i \in \sigma_\theta} R^i_\theta$ and factor $R_2 = \bigcap_{\theta \in B-C} \bigcup_{i \in \sigma_\theta} R^i_\theta$. The point of this is that $|R_1 \cap R_2| = |R_1| \times |R_2|$ because no i appears in both expressions R_1 and R_2. We can recursively use the method (using simplifying, splitting and factoring) to compute $|R_1|$ and $|R_2|$, and compute the volume of R as $|R_1| \times |R_2|$.

5.3.2 An algebraic view of the approach

We briefly sketch another view of the approach of 5.3.1, based on rearranging multiple summations of mass functions. Let m_i be the mass function associated with the ith source triple, so that, for $C \subseteq \Theta$, $\mathrm{m}_i(C) = \mathrm{P}_i(\Gamma_i(\omega_i) = C)$. The term $\mathrm{P}'(\Gamma'(\omega) \subseteq A)$ can be shown to be equal to

$$\sum_{\substack{A_1, \ldots, A_k : \\ A_1 \cap \cdots \cap A_k \subseteq A}} \mathrm{m}_1(A_1) \cdots \mathrm{m}_k(A_k).$$

For example, Splitting by k corresponds to writing this multiple summation as

$$\sum_{A_k \,:\, \mathrm{m}_k(A_k) \neq 0} \mathrm{m}_k(A_k)\; S^k_{A_k}.$$

Here, $S^k_{A_k}$ is the multiple summation

$$\sum \mathrm{m}_1(A_1) \cdots \mathrm{m}_{k-1}(A_{k-1})$$

where the summation is over all sequences of sets $A_1, \ldots, A_{k-1}$ such that their intersection $\bigcap_{i=1}^{k-1} A_i$ is a subset of $A \cup \overline{A_k}$.

The $|\mathcal{F}_{\mathrm{m}_k}|$ internal summations can then be recursively split until the summations can be easily evaluated.

6 Approximate Methods for Calculating Combined Belief

Because of the computational problems of exact combination of Dempster-Shafer belief, it is natural to consider approximate techniques. In this section, a number of such techniques are briefly discussed.

6.1 Bayesian Approximation

Define the Bayesian approximation [Voorbraak, 89; see also Joshi *et al*, 95] $\underline{\mathrm{m}}$ of a mass function m to be Bayesian, i.e., all its focal sets are singletons, and for $\theta \in \Theta$,

$$\underline{\mathrm{m}}(\{\theta\}) = \lambda \mathrm{Q}_{\mathrm{m}}(\{\theta\}) = \lambda \sum_{A \ni \theta} \mathrm{m}(A),$$

where the normalising constant λ is given by

$$\lambda^{-1} = \sum_{\theta \in \Theta} \mathrm{Q}_{\mathrm{m}}(\{\theta\}) = \sum_{A \subseteq \Theta} \mathrm{m}(A)|A|.$$

The term λ^{-1} can thus be viewed as the mass-weighted average of the sizes of the focal sets.

Naturally, the Bayesian approximation $\underline{\mathrm{Bel}}$ of belief function Bel is then defined to be the belief function associated with $\underline{\mathrm{m}}$, where m is the mass function associated with Bel; we can also define $\underline{\mathrm{Pl}}$ in an analogous way; we have $\underline{\mathrm{Pl}} = \underline{\mathrm{Bel}}$.

A nice property of Bayesian approximation is that the operation commutes with combination: the Bayesian approximation of the combination of a number of mass functions $\mathrm{m}_1, \ldots, \mathrm{m}_k$ is the same as the combination of the Bayesian approximations of the mass functions, i.e., $\underline{\mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k}$ equals $\underline{\mathrm{m}_1} \oplus \cdots \oplus \underline{\mathrm{m}_k}$.

This follows easily from the simple multiplicative form of Dempster's rule for commonalities (see section 2.4).

The computation of the combination of the Bayesian approximations can be performed quickly, in time proportional to $|\Theta|$, and this can be used to approximate the combination of the input mass functions. The key issue is how good an approximation it is. We'll first consider its use for approximating the combined plausibility, and then for approximating combined belief.

Let Pl be the plausibility function associated with $\mathrm{m} = \mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_k$, and let $\underline{\mathrm{Pl}}$ be the plausibility function associated with $\underline{\mathrm{m}}$. For singleton set $\{\theta\}$, with $\theta \in \Theta$, $\mathrm{Pl}(\{\theta\}) = \mathrm{Q}(\{\theta\})$, and,

$$\underline{\mathrm{Pl}}(\{\theta\}) = \underline{\mathrm{m}}(\{\theta\}) = \frac{\mathrm{Q}(\{\theta\})}{\sum_{\psi \in \Theta} \mathrm{Q}(\{\psi\})}$$

where $\mathrm{Q} = \mathrm{Q}_{\mathrm{m}}$ is the commonality function associated with m.

We thus have

$$\frac{\mathrm{Pl}(\{\theta\})}{\underline{\mathrm{Pl}}(\{\theta\})} = \lambda^{-1} = \sum_{\psi \in \Theta} \mathrm{Q}(\{\psi\}) = \sum_{A \subseteq \Theta} \mathrm{m}(A)|A|,$$

the mass-weighted average of the sizes of the focal sets. This lies between 1 and $|\Theta|$. It is clear, then, that even for singletons it can be a very poor approximation, out by a factor of 2 even if the average size of the focal sets (of m) is as small as 2. Note, however, that the Bayesian approximation does give correct values for the relative values of plausibilities of singletons: for $\theta, \psi \in \Theta$,

$$\frac{\mathrm{Pl}(\{\theta\})}{\mathrm{Pl}(\{\psi\})} = \frac{\underline{\mathrm{Pl}}(\{\theta\})}{\underline{\mathrm{Pl}}(\{\psi\})}.$$

Also, for singleton sets $\{\theta\}$, $\underline{\mathrm{Pl}}(\{\theta\})$ is not larger than $\mathrm{Pl}(\{\theta\})$; this result does not generalise to arbitrary sets $A \subseteq \Theta$, as the following example shows[11]:

Label Θ as $\{\theta_1, \ldots, \theta_n\}$ and let $A = \{\theta_1, \ldots, \theta_{n-1}\}$. Define mass function m by $\mathrm{m}(A) = \frac{1}{n}$ and $\mathrm{m}(\{\theta_n\}) = \frac{n-1}{n}$. Then we have $\mathrm{Pl}(A) = \frac{1}{n}$ and $\underline{\mathrm{Pl}}(A) = \frac{1}{2}$ so $\mathrm{Pl}(A)/\underline{\mathrm{Pl}}(A) = \frac{2}{n}$ which is close to 0 for large n.

$\underline{\mathrm{Bel}}$ also often approximates Bel poorly. If $\{\theta\}$ is not a focal set then $\mathrm{Bel}(\{\theta\})$ will be 0; however the Bayesian approximation will give a non-zero value to $\{\theta\}$ so long as there is some focal set containing θ; an extreme example is a vacuous belief function (so that $\mathrm{Bel}(A) = 1$ iff $A = \Theta$ and otherwise $\mathrm{Bel}(A) = 0$) over a frame Θ of two elements $\{\theta, \psi\}$. We then have $\mathrm{Bel}(\{\theta\}) = 0$ but $\underline{\mathrm{Bel}}(\{\theta\}) = \frac{1}{2}$.

Clearly what is crucial for the Bayesian approximation to be a good approximation of combined belief (or plausibility) is if the combination is very close to being Bayesian. This *can* easily happen, but it is certainly not necessarily the case even for very large k (the relative sizes of k and $|\Theta|$ are often relevant).

[11] This means that the Bayesian approximation of a belief function Bel is not in the set of compatible measures $\mathcal{P}$ associated with Bel (see section 8.1).

6.2 Storing Fewer Focal Sets

Another approximate method, suggested by [Tessem 93], is to use the basic iterative algorithm for combining with Dempster's rule (see section 5.1), but, at each stage, if the number of focal sets becomes too large, to store just some of them, for example, those with largest mass.

Although it appears from the experimental results of [Tessem, 93] that this can sometimes yield reasonable results, there is an obvious very serious problem. It appears that the number of focal sets of the combination of k mass functions over Θ will tend to grow exponentially in the minimum of $|\Theta|$ and k. In which case, either such a method will be exponential (if we keep a sizeable proportion of the focal sets) or we'll only keep a very small proportion of the focal sets, in which case it seems unlikely that we'll usually be able to keep a representative set. The related approaches described in [Bauer, 97] suffer from the same problem.

For example, let $\Theta = \{\theta_1, \ldots, \theta_{20}\}$, and, for $i = 1, \ldots, 20$, define m_i by $\mathrm{m}_i(\Theta - \{\theta_i\}) = \frac{1}{3}$ and $\mathrm{m}_i(\Theta) = \frac{2}{3}$. There are more than a million focal sets of the combined mass function $\mathrm{m} = \mathrm{m}_1 \oplus \cdots \oplus \mathrm{m}_{20}$. Even if we were storing at each stage the set $\mathcal{F}'$ of 1000 focal sets with largest masses, at the end of the calculation, the total mass covered is less than 0.05, i.e., $\sum_{A \in \mathcal{F}'} \mathrm{m}(A) < 0.05$. For larger k and $|\Theta|$ the situation gets much worse.

It therefore seems unlikely that this type of method could generally yield good approximations of values of Bel when k and $|\Theta|$ are not small.

6.3 Using Direct Computation of Belief Algorithm to Give Bounds

The algorithm of 5.3.1 involves recursively breaking down an expression (representing a region in a multi-dimensional space) into simpler ones until their hyper-volumes can be calculated. At each stage, we can easily compute an upper bound for the contribution an expression will make to the volume of the original expression (and 0 is a lower bound). In this way the algorithm can be used to give upper and lower bounds for the volume of region R, which are improved as the algorithm progresses, with the upper and lower bounds eventually becoming equal when the algorithm is complete. However, the volumes of some expressions are much harder to calculate than others, so if we wish, we could terminate the algorithm early, without computing all the volumes, and return the lower and upper bounds for the volume of R.

For example if we split R by l, slicing R into disjoint regions $\{R_{\omega_l} : \omega_l \in \Omega_l\}$ it may happen that the volume of one of these sub-regions, say $R_{\omega_l^1}$ is hard to compute, but we find that all the others are easily calculated, and they sum to (say) 0.4. If $\mathrm{P}_l(\omega_l^1)$ is, say, 0.05, then the volume of R must then lie in the interval $[0.4, 0.45]$.

Recall that the volume of R is $\mathrm{P}'(\Gamma'(\omega) \subseteq A)$. In the same way we can get

upper and lower bounds for $\mathrm{P}'(\Gamma'(\omega) = \emptyset)$, and use the two sets of bounds to get upper and lower bounds for $\mathrm{Bel}(A)$.

However, this method may suffer from similar problems to the previous one (6.2): it may not be possible to get a good approximation of belief in a reasonable amount of time.

6.4 Other Approximate Methods

[Gordon and Shortliffe, 85] considers the problem of combining simple support functions when the focal sets are either members of, or complements of members of, a hierarchical hypothesis space. The latter is a set of subsets of the frame containing $\emptyset$ and such that for any two sets in it, their intersection is either empty or one of the sets. Earlier [Barnett, 81] had successfully tackled the simpler problem where the focal sets of the simple support functions are either singletons or complements of singletons (or, of course, the frame Θ). Gordon and Shortliffe used an approximation for their problem; however, the approximation is not always a good one (see [Wilson, 87]), and, in any case, combined belief can be calculated efficiently exactly [Shafer and Logan, 87; Wilson, 87; 89].

[Dubois and Prade, 90] discuss the approximation of an arbitrary belief functions by a consonant belief function, i.e., one whose focal sets are nested. However, as pointed out in [Tessem, 93], this approximation is not well suited to the calculation of combined belief.

7 Monte-Carlo Algorithms for Calculating Combined Belief on Frame

It seems that the exact methods described above (section 5) are probably only useful for relatively small problems. An alternative approach is to use Monte-Carlo algorithms; combined Dempster-Shafer belief is estimated using a large number L of trials of a random algorithm; each trial gives an estimate of belief (a very poor one: either 0 or 1); an average of the estimates of all the trials converges to the correct value of belief as L gets large. In this way, much faster algorithms for finding combined belief (up to a given degree of accuracy) are possible, almost linear in the size of the frame for the algorithms given in 7.3 and 7.4.1.

It is assumed that we're interested in calculating values of combined belief. However, since there are exponentially many subsets of Θ, for large Θ it will not be feasible to calculate the belief in all of them. Instead, it is assumed that, for a fairly small number of important sets $A \subseteq \Theta$, we are interested in calculating $\mathrm{Bel}(A)$. To simplify the presentation of the algorithms we just consider one set A, but the extension to a number of sets is straight-forward.

The algorithms involve random sampling so it is natural to express them in terms of source triples, which separate out the underlying probability distribution. If we are given mass functions to combine, we just have to convert each to a source triple (which is associated with the mass function; see section 2.2). For the reader's convenience the basic definitions given in 2.2 and 2.4.2 are repeated here.

A source triple over Θ is a triple $(\Omega, \mathrm{P}, \Gamma)$ where Ω is a finite set, P is a strictly positive probability distribution over Ω and Γ is a function from Ω to $2^\Theta - \{\emptyset\}$. Associated with a source triple is a mass function, and hence a belief function, given respectively by $\mathrm{m}(A) = \sum_{\omega\,:\,\Gamma(\omega)=A} \mathrm{P}(\omega)$ and $\mathrm{Bel}(A) = \sum_{\omega\,:\,\Gamma(\omega)\subseteq A} \mathrm{P}(\omega)$. Dempster's rule for source triples is a mapping sending a finite set of source triples $\{(\Omega_i, \mathrm{P}_i, \Gamma_i)$, for $i = 1, \ldots, k\}$, to a triple $(\Omega, \mathrm{P}_{\mathrm{DS}}, \Gamma)$, defined as follows. Let $\Omega^\times = \Omega_1 \times \cdots \times \Omega_k$. For $\omega \in \Omega^\times$, $\omega(i)$ is defined to be its ith component (sometimes written ω_i), so that $\omega = (\omega(1), \ldots, \omega(k))$. Define $\Gamma' \colon \Omega^\times \to 2^\Theta$ by $\Gamma'(\omega) = \bigcap_{i=1}^k \Gamma_i(\omega(i))$ and probability distribution P' over $\Omega^\times$ by $\mathrm{P}'(\omega) = \prod_{i=1}^k \mathrm{P}_i(\omega(i))$, for $\omega \in \Omega^\times$. Let Ω be the set $\{\omega \in \Omega^\times : \Gamma'(\omega) \neq \emptyset\}$, let Γ be Γ' restricted to Ω, and let probability distribution P_{DS} over Ω be P' conditioned on Ω, so that for $\omega \in \Omega$, $\mathrm{P}_{\mathrm{DS}}(\omega) = \mathrm{P}'(\omega)/\mathrm{P}'(\Omega)$. The factor $1/\mathrm{P}'(\Omega)$ can be viewed as a measure of the conflict between the evidences.

The combined measure of belief Bel over Θ is given, for $A \subseteq \Theta$, by $\mathrm{Bel}(A) = \mathrm{P}_{\mathrm{DS}}(\{\omega \in \Omega : \Gamma(\omega) \subseteq A\})$, which we abbreviate to $\mathrm{P}_{\mathrm{DS}}(\Gamma(\omega) \subseteq A)$.

As usual, it can be helpful to pre-process the source triples by conditioning each by their combined core (see 3.3). Lossless coarsening (see 3.4) can also sometimes be helpful.

7.1 A Simple Monte-Carlo Algorithm

Dempster's set-up [Dempster, 67] suggests a natural Monte-Carlo algorithm for calculating belief (or other related quantities). Section 7.1 is based on [Wilson, 91], but variations of the same idea are given in [Kämpke, 88; Pearl, 88; Wilson, 89; Kreinovich *et al.*, 92].

Since, for $A \subseteq \Theta$, $\mathrm{Bel}(A) = \mathrm{P}_{\mathrm{DS}}(\Gamma(\omega) \subseteq A)$, to calculate $\mathrm{Bel}(A)$ we can repeat a large number of trials of a Monte-Carlo algorithm where for each trial, we pick ω with chance $\mathrm{P}_{\mathrm{DS}}(\omega)$ and say that the trial succeeds if $\Gamma(\omega) \subseteq A$, and fails otherwise. $\mathrm{Bel}(A)$ is then estimated by the proportion of the trials that succeed. The random algorithms described in 7.1, 7.2 and 7.3 all involve simulating P_{DS}. The most straight-forward way is to pick ω with chance $\mathrm{P}_{\mathrm{DS}}(\omega)$ by repeatedly (if necessary) picking $\omega \in \Omega^\times$ with chance $\mathrm{P}'(\omega)$ until we get an ω in Ω. Picking ω with chance $\mathrm{P}'(\omega)$ is easy: for each $i = 1, \ldots, k$, we pick $\omega_i \in \Omega_i$ with chance $\mathrm{P}_i(\omega_i)$ and let $\omega = (\omega_1, \ldots, \omega_k)$.

For each trial:

for $i = 1, \ldots, k$

pick $\omega_i \in \Omega_i$ with chance $\mathrm{P}_i(\omega_i)$
let $\omega = (\omega_1, \ldots, \omega_k)$
if $\Gamma(\omega) = \emptyset$ (so $\omega \notin \Omega$)
then restart trial
else if $\Gamma(\omega) \subseteq A$
then trial succeeds (i.e., gets the value 1)
else trial fails (i.e., gets the value 0)

The time that the algorithm takes to achieve a given accuracy (with a given high probability) is roughly proportional[12] to $|\Theta|k/\mathrm{P}'(\Omega)$, making it very efficient for problems where the evidences are not very conflicting [Wilson 91]. The reason for this efficiency is that the number of (completed) trials needed to achieve a given accuracy is not dependent on the size of the problem (e.g., $|\Theta|$ and k).

If the belief functions are highly conflicting, so that $\mathrm{P}'(\Omega)$ is extremely small, then it will tend to take a very long time to find an ω in Ω, as illustrated by the following example.

Example Let $\Theta = \{x_1, x_2, \ldots, x_k\}$, for each $i = 1, \ldots, k$, let $\Omega_i = \{1, 2\}$, let $\mathrm{P}_i(1) = \mathrm{P}_i(2) = \frac{1}{2}$, let $\Gamma_i(1) = \{x_i\}$ and let $\Gamma_i(2) = \Theta$. The triple $(\Omega_i, \mathrm{P}_i, \Gamma_i)$ corresponds to a simple support function with $\mathrm{m}_i(\{x_i\}) = \frac{1}{2}$ and $\mathrm{m}_i(\Theta) = \frac{1}{2}$. The conflict between the evidences is very high for large k since we have $\mathrm{P}'(\Omega) = (k+1)/2^k$ so the simple Monte-Carlo algorithm is not practical.

7.2 A Markov Chain Algorithm

We will consider Monte-Carlo algorithms where the trials are not independent, but instead form a Markov Chain, so that the result of each trial is (probabilistically) dependent only on the result of the previous trial. Section 7.2 is an edited version of [Moral and Wilson, 94].

Both this and the random algorithm using commonality (section 7.3) are based on Markov Chains that simulate P_{DS}, that is, a sequence $\omega^0, \omega^1, \ldots, \omega^L$ of elements in Ω, (in these two algorithms, starting element ω^0 can be chosen arbitrarily), where ω^l depends randomly on ω^{l-1}, but not on any previous member of the sequence, and that for sufficiently large L, ω^L will have distribution very close to $\mathrm{P}_{\mathrm{DS}}(\cdot)$. As for the simple algorithm we can then test if $\Gamma(\omega^L) \subseteq A$; this procedure is repeated sufficiently many times to get a good estimate of $\mathrm{Bel}(A)$.

7.2.1 The Connected Components of Ω

The Markov Chain algorithms of section 7.2 require a particular condition on Ω to work, which we will call connectedness. This corresponds to the Markov

[12] Naturally, the constant of proportionality is bigger if greater accuracy is required.

Chain being irreducible [Feller, 50].

For $i \in \{1, \ldots, k\}$ and $\omega, \omega' \in \Omega$ write $\omega \equiv_i \omega'$ if ω and ω' differ at most on their ith co-ordinate, i.e., if for all $j \in \{1, \ldots, k\} - \{i\}$, $\omega(j) = \omega'(j)$. Let U be the union of the relations $\equiv_i$ for $i \in \{1, \ldots, k\}$, so that $\omega \; U \; \omega'$ if and only if ω and ω' differ at most on one co-ordinate; let equivalence relation $\equiv$ be the transitive closure of U. The equivalence classes of $\equiv$ will be called *connected components* of Ω, and Ω will be said to be *connected* if it has just one connected component, i.e, if $\equiv$ is the relation $\Omega \times \Omega$.

A method for testing connectedness is sketched in section 5.1 of [Moral and Wilson, 94]; the number of operations needed is at worst proportional to $|\Theta|^2 \sum_{i=1}^{k} |\Omega_i|$.

7.2.2 The Basic Markov Chain Monte-Carlo Algorithm

Probabilistic function $\mathrm{PDS}^L(\omega^0)$ takes as input initial state $\omega^0 \in \Omega$ and number of trials L and returns a state ω. The intention is that when L is large, for any initial state ω^0, $\Pr(\mathrm{PDS}^L(\omega^0) = \omega)$ is very close to $\mathrm{P}_{\mathrm{DS}}(\omega)$ for all $\omega \in \Omega$. The algorithm starts in state ω^0 and randomly moves between elements of Ω. The initial state ω^0 can be picked arbitrarily; one way of doing this is to pick arbitrarily an element θ in the combined core, and then choose, for each $i = 1, \ldots, k$, some $\omega_i \in \Omega_i$ such that $\Gamma_i(\omega_i) \ni \theta$; we can then let ω^0 equal $(\omega_1, \ldots, \omega_k)$, which is in Ω because $\Gamma(\omega) \ni \theta$.

In the algorithms the current state is labelled ω^c.

FUNCTION $\mathrm{PDS}^L(\omega^0)$
  $\omega^c := \omega^0$
  for $l = 1$ **to** L
    for $i = 1$ **to** k
      ω^c **:=** $\mathit{operation}_i(\omega^c)$
    next i
  next l
  return ω^c**.**

Probabilistic function $\mathit{operation}_i$ changes at most the ith co-ordinate of its input ω^c—it changes it to y with chance proportional to $\mathrm{P}_i(y)$. We therefore have, for $\omega, \omega' \in \Omega$,

$$\Pr(\mathit{operation}_i(\omega') = \omega) = \begin{cases} \alpha_{\omega'} \mathrm{P}_i(\omega(i)) & \text{if } \omega \equiv_i \omega'; \\ 0 & \text{otherwise.} \end{cases}$$

The normalisation constant $\alpha_{\omega'}$ is given by $\alpha_{\omega'}^{-1} = \sum_{\omega \equiv_i \omega'} \mathrm{P}_i(\omega(i))$.

The reason that we require that Ω be connected is that that, in the algorithms, the only values that ω^c can take are the members of the $\equiv$-equivalence class of the starting position ω^0.

7.2.3 The Calculation of Belief

If Ω is connected, then for sufficiently large L (and any choice of ω^0), $\mathrm{PDS}^L(\omega^0)$ will have distribution very close to $\mathrm{P_{DS}}$. We can use this in a natural way to produce algorithms for calculating $\mathrm{Bel}(A)$. Function $\mathrm{B}_J^L(\omega^0)$ has inputs ω^0, L and J, where $\omega^0 \in \Omega$ is a starting value, L is the number of trials, and J is the number of trials used by the function $\mathrm{PDS}^J(\cdot)$ used in the algorithm. The value $\mathrm{B}_J^L(\omega^0)$ can be seen to be the proportion of the L trials in which $\Gamma(\omega^c) \subseteq A$.

In the $\mathrm{B}_J^L(\omega^0)$ algorithm, for each call of $\mathrm{PDS}^J(\cdot)$, Jk values of ω are generated, but only one, the last, is used to test if $\Gamma(\omega) \subseteq A$. Alternatively, all of the values could be used, which is what $\mathrm{BEL}^L(\omega^0)$ does. The implementation is very similar to that for $\mathrm{PDS}^L(\omega^0)$, the main difference being the extra **if**-statement in the inside **for** loop. The value returned by $\mathrm{BEL}^L(\omega^0)$ is the proportion of the time that $\Gamma(\omega^c) \subseteq A$.

FUNCTION $\mathbf{B}_J^L(\omega^0)$
$\omega^c := \omega^0$
$S := 0$
for $l = 1$ **to** L
  $\omega^c := \mathbf{PDS}^J(\omega^c)$
  if $\Gamma(\omega^c) \subseteq A$
    then $S := S + 1$
next l
return $\frac{S}{L}$

FUNCTION $\mathbf{BEL}^L(\omega^0)$
$\omega^c := \omega^0$
$S := 0$
for $l = 1$ **to** L
  for $i = 1$ **to** k
    ω^c **:=** $operation_i(\omega^c)$
    if $\Gamma(\omega^c) \subseteq A$
      then $S := S + 1$
  next i
next l
return $\frac{S}{Lk}$

Although the algorithms are guaranteed to converge to the correct value (for connected Ω) it is not clear how quickly this will happen. The following example illustrates that the convergence rate will tend to be very slow if Ω is only barely connected, i.e., if it is very hard for the algorithm to move between some elements of Ω. For $\theta \in \Theta$, define $\theta^* \subseteq \Omega$ to be the set $\{\omega \in \Omega : \Gamma(\omega) \ni \theta\}$.

Example

Let $k = 2q - 1$, for some $q \in \mathbb{N}$, and let $\Theta = \{x_1, x_2\}$. For each $i = 1, \ldots, k$, let $\Omega_i = \{1, 2\}$, let $\mathrm{P}_i(1) = \mathrm{P}_i(2) = \frac{1}{2}$, let $\Gamma_i(2) = \Theta$ and, for $i \leq q$, let $\Gamma_i(1) = \{x_1\}$, and, for $i > q$, let $\Gamma_i(1) = \{x_2\}$. Each triple $(\Omega_i, \mathrm{P}_i, \Gamma_i)$ corresponds to a simple support function. Ω is very nearly not connected since it is the union of two sets $\{x_1\}^*$ (which has 2^q elements) and $\{x_2\}^*$ (which has 2^{q-1} elements) which have just a singleton intersection $\{(2, \ldots, 2)\}$.

Suppose we want to use function $\mathrm{B}_J^L(\omega^0)$ or function $\mathrm{BEL}^L(\omega^0)$ to estimate $\mathrm{Bel}(\{x_1\})$ (which is just under $\frac{2}{3}$). If we start with ω^0 such that $\Gamma(\omega^0) = \{x_1\}$

then it will probably take of the order of 2^q values of ω to reach a member of $\{x_2\}^*$. Therefore if q is large, e.g. $q = 30$, and we do a million trials then our estimate of $\mathrm{Bel}(\{x_1\})$ will almost certainly be 1. Other starting positions ω^0 have similar problems.

Since $\mathrm{P}'(\Omega) \approx 3/2^q$ the simple Monte-Carlo algorithm does not perform satisfactorily here either. (If Ω is barely connected, then it will usually be small in comparison to $\Omega^\times$, so the contradiction will tend to be high, and the simple Monte-Carlo algorithm will not work well either.)

Some possible ways of trying to solve this problem are suggested in section 8 of [Moral and Wilson, 94].

It may not be immediately obvious if the estimate of belief is close to convergence or not; in the above example it would appear that the estimate of $\mathrm{Bel}(\{x_1\})$ has converged to 1, when in fact it's far from convergence. There is a way of seeing if the algorithm isn't close to convergence: compute exactly the relative commonalities of singletons, more precisely, $\mathrm{Q}(\{\theta_j\})/\mathrm{Q}(\{\theta_h\})$ for all $\theta_j \in \Theta$, where $\mathrm{Q}(\{\theta_h\})$ is the largest of these values—this can be done using the simple form of Dempster's rule for commonalities; also use the generated values of ω^c to estimate these ratios. If there is any substantial difference, then the algorithm isn't close to convergence.

7.3 A Random Algorithm Using Commonality

Let $\underline{\Theta}$ be the core of the combined mass function, i.e., the union of its focal sets, which can be calculated as the intersection of the cores of the constituent mass functions (see section 3.3) $\mathcal{C}_{\mathrm{m}_1} \cap \cdots \cap \mathcal{C}_{\mathrm{m}_k}$ where $\mathcal{C}_{\mathrm{m}_i} = \bigcup_{\omega_i \in \Omega_i} \Gamma_i(\omega_i)$. Of course if, before applying the algorithms, we have conditioned all the source triples by the combined core, and redefined Θ to be this, then we will have $\underline{\Theta} = \Theta$.

As in section 7.2, we want to generate a Markov chain that converges rapidly to P_{DS}; this can then be used to pick an element ω in Ω with chance approximately $\mathrm{P}_{\mathrm{DS}}(\omega)$; we can then test if $\Gamma(\omega) \subseteq A$, and we can repeat the process until we have a high probability of being within a small range of the correct value of $\mathrm{Bel}(A)$.

Recall that for $\theta \in \Theta$, the set $\theta^* = \{\omega \in \Omega : \Gamma(\omega) \ni \theta\}$. Although we can't always sample efficiently with P_{DS} over the whole of Ω, we can sample easily within each θ^*, because it's a product subset of Ω: $\theta^* = \prod_{i=1}^k \theta_i^*$, where θ_i^* is defined to be $\{\omega_i \in \Omega_i : \Gamma_i(\omega_i) \ni \theta\}$; $\mathrm{P}_{\mathrm{DS}}(\omega|\theta^*) = \mathrm{P}'(\omega|\theta^*) = \prod_{i=1}^k \mathrm{P}_i(\omega(i)|\theta_i^*)$, so we can pick a random element ω of θ^* by picking random element ω_i of θ_i^*, for each $i = 1, \ldots, k$, and letting $\omega = (\omega_1, \ldots, \omega_k)$.

Furthermore, Ω is equal to the union of regions θ^* for $\theta \in \underline{\Theta}$, and we can easily pick region θ^* with chance proportional to $\mathrm{P}_{\mathrm{DS}}(\theta^*)$: define the unnormalised commonality function Q' by, for $A \subseteq \Theta$, $Q'(A) = \prod_{i=1}^k Q_i(A)$ where $Q_i(A) = \mathrm{P}_i(\Gamma_i(\omega_i) \supseteq A)$ (Q_i is the commonality function associated with the

ith input source triple.) Define probability distribution $Q'' : \underline{\Theta} \to [\,0,1\,]$ by setting, for $\theta \in \underline{\Theta}$, $Q''(\theta) = Q'(\{\theta\}) / \sum_{\psi \in \underline{\Theta}} Q'(\{\psi\})$. It can be shown that $Q''(\theta)$ is proportional to $\mathrm{P}_{\mathrm{DS}}(\theta^*)$.

Hence the idea is to randomly pick a $\theta \in \underline{\Theta}$ and then randomly pick an element $\omega \in \theta^*$. Element θ is picked with chance proportional to $\mathrm{P}_{\mathrm{DS}}(\theta^*)$, i.e., with chance Q'', and ω is then picked with chance $\mathrm{P}'(\omega|\theta^*)$.

However this scheme is biased towards ω with large $\Gamma(\omega)$, since there are $|\Gamma(\omega)| = |\{\theta : \theta^* \ni \omega\}|$ ways of reaching ω (each of them equally likely). To correct this bias we arrange that, if ω^{l-1} is the state after $l-1$ trials, there is only a chance proportional to $|\Gamma(\omega^{l-1})|$ that a new value is picked; otherwise nothing happens (i.e., ω^l is set to ω^{l-1}).

Bringing these parts together (for other technical points, see [Wilson and Moral, 96]) gives the following algorithm:

for $l = 1$ **to** L
  if RND() $\geq \frac{0.9}{|\Theta|} |\Gamma(\omega^{l-1})|$
    then $\omega^l := \omega^{l-1}$
    else
      Pick $\theta \in \underline{\Theta}$ **with distribution** Q''
      Pick $\omega^l \in \theta^*$ **with distribution** $\mathrm{P}'(\cdot|\theta^*)$
  end if
next l

where RND() is a random number generator, taking a random value in $[\,0,1\,]$ with uniform distribution each time it is called.

This Markov chain converges in a fairly predicable way, and can be used to calculate $\mathrm{Bel}(A)$ in a similar way to that described in the previous section (7.2). The overall computation is approximately proportional to $|\underline{\Theta}|$ and k^2 (or possibly k); see [Wilson and Moral, 96] for details.

7.4 Importance Sampling Algorithms

The idea behind these importance (or weighted) sampling algorithms of Serafín Moral [Moral and Wilson, 96] is instead of sampling with the distribution of interest P_{DS}, we sample with a different distribution P^* which is easier to sample with, and then assign a weight proportional to $\mathrm{P}_{\mathrm{DS}}(\omega)/\mathrm{P}^*(\omega)$ to each trial. We can then estimate $\mathrm{Bel}(A)$, for $A \subseteq \Theta$, by calculating the sum of weights for all trials such that $\Gamma(\omega) \subseteq A$, and dividing this by the sum of weights for all trials.

7.4.1 Commonality-based importance sampling

The algorithm described in 7.3 can spend a lot of time hanging around doing nothing (i.e., we very often have ω^l set to ω^{l-1}). It will sometimes be much more efficient to weight the trials instead, which is what the following algorithm does.

$S := 0$
$S' := 0$
for $l = 1$ **to** L
  Pick $\theta \in \underline{\Theta}$ **with distribution** Q''
  Pick $\omega \in \theta^*$ **with distribution** $\mathrm{P}'(\cdot|\theta^*)$
  $W := 1/|\Gamma(\omega)|$
  if $\Gamma(\omega) \subseteq A$ **then**
    $S := S + W$
    else $S' := S' + W$
next l
return $\frac{S}{S+S'}$

The probability that particular value ω is picked in a given trial of the above algorithm is $\sum_{\theta \in \underline{\Theta}} Q''(\theta)\mathrm{P}'(\omega|\theta^*)$ which is equal to $\mathrm{P}_{\mathrm{DS}}(\omega)|\Gamma(\omega)|/e$ where $e = \sum_{\theta \in \Theta} \mathrm{Q}(\{\theta\}) = \sum_{A \subseteq \Theta} \mathrm{m}(A)|A|$, where Q and m are the combined (normalised) commonality and mass functions respectively. e can also seen to be the expected value of $|\Gamma(\omega)|$ with respect to P_{DS}.

Let $T = S + S'$. The expected value of eS/L is equal to $\mathrm{Bel}(A)$ and the expected value of eT/L equals 1. The variance of eS/L is less than $\mathrm{Bel}(A)e/L$ and that of eT/L is less than e/L. Therefore the expected value of random variable S/T (the ratio of eS/L and eT/L) tends to $\mathrm{Bel}(A)$ as the number of trials L tends to ∞. The number of trials needed for the algorithm to achieve a given accuracy in its estimate of $\mathrm{Bel}(A)$ (with a given confidence level) is at worst linear in e and hence at worst linear in $|\underline{\Theta}|$ (but typically substantially sublinear). The preprocessing and time required for each trial is similar to the algorithm in 7.3 (see [Wilson and Moral, 96]). Overall the algorithm is approximately linear in $k|\underline{\Theta}|$.

7.4.2 Importance sampling based on consistency

For the following algorithm to work we need that each $\omega_i \in \Omega_i$ is consistent with Ω (i.e., for each $\omega_i \in \Omega_i$ there exists some $\omega \in \Omega$ with $\omega(i) = \omega_i$), or equivalently, that each focal set of any of the input mass functions is consistent with the combined core. One way of ensuring this is to compute the combined core and condition each input source triple by this, before proceeding further.

The simple Monte-Carlo algorithm (see 7.1) was based on picking ω with chance $\mathrm{P}'(\omega)$ and afterwards, checking whether ω is in Ω, that is, if $\bigcap_{i=1}^{k} \Gamma_i(\omega_i) \neq \emptyset$; if not, then ω is repicked. The selection of ω was carried out in the following way: for each $i = 1, \ldots, k$, we picked $\omega_i \in \Omega_i$ with chance $\mathrm{P}_i(\omega_i)$ and let $\omega = (\omega_1, \ldots, \omega_k)$. We can avoid this continual repicking of ω if we ensure that when co-ordinate ω_i is picked, the choice is consistent with previous choices, i.e., that $\bigcap_{j=1}^{i} \Gamma_j(\omega_j) \neq \emptyset$. Let Δ_i^ω be the set of consistent choices for the ith co-ordinate, i.e., $\{\omega_i \in \Omega_i : \bigcap_{j=1}^{i} \Gamma_j(\omega_j) \neq \emptyset\}$, and let $C_i^\omega = \mathrm{P}_i(\Delta_i^\omega)$. As before, we pick ω_i in Δ_i^ω, with chance proportional to $\mathrm{P}_i(\omega_i)$; hence ω_i is picked with chance $\mathrm{P}_i(\omega_i)/C_i^\omega$.

This biases the probability distribution away from P_{DS} so we need to assign a weight $\prod_{i=1}^{k} C_i^\omega$ to each trial to compensate (where ω is the random element of Ω which gets picked).

Hence we have the following algorithm to estimate $\mathrm{Bel}(A)$.

$S := 0$
$S' := 0$
for $l = 1$ **to** L
  $W := 1$
  for $i = 1$ **to** k
    $W := W * C_i^\omega$
    pick $\omega_i \in \Delta_i^\omega$ **with chance** $\mathrm{P}_i(\omega_i)/C_i^\omega$
  next i
  $\omega := (\omega_1, \ldots, \omega_k)$
  if $\Gamma(\omega) \subseteq A$ **then**
    $S := S + W$
    else $S' := S' + W$
next l
return $\frac{S}{S+S'}$

As L tends to infinity, the expected value of $\frac{S}{S+S'}$ tends to $\mathrm{Bel}(A)$; however the variance and the rate of convergence don't seem to be easy to generally determine.

A development of consistency-based importance sampling

A development of this algorithm has recently been proposed in [Moral and Salmerón, 99]. As above, elements $\omega_1, \omega_2, \ldots$, are picked in turn subject to the constraint that $\omega_i \in \Delta_i^\omega$, but with different chances. Suppose we have picked $\omega_1, \omega_2, \ldots, \omega_{i-1}$. Let Ω'_{ω_i} be the set of possible ω which the algorithm can end up picking (from this point), given that we pick ω_i next, i.e., $\Omega'_{\omega_i} = \{\omega \in \Omega : \text{ for all } j = 1, \ldots, i, \omega(j) = \omega_j\}$. Since the aim is to simulate P_{DS}, to

reduce the variance, ideally we would like to choose $\omega_i \in \Delta_i^\omega$ with chance proportional to $\mathrm{P}_{\mathrm{DS}}(\Omega'_{\omega_i})$, i.e., proportional to $\mathrm{P}'(\Omega'_{\omega_i})$. Now, Ω'_{ω_i} is the set of all $\omega \in \Omega^\times$ such that (i) for $j = 1, \ldots, i$, $\omega(j) = \omega_j$ and (ii) $\bigcap_{j>i} \Gamma_j(\omega_j) \cap (X \cap \Gamma_i(\omega_i)) \neq \emptyset$, where $X = \bigcap_{j<i} \Gamma_j(\omega_j)$ is the intersection of the focal sets picked so far. Therefore $\mathrm{P}'(\Omega'_{\omega_i})$ is proportional to $\left(\prod_{j<i} \mathrm{P}_j(\omega_j)\right) \mathrm{P}_i(\omega_i)\mathrm{Pl}_{>i}(X \cap \Gamma_i(\omega_i))$, where $\mathrm{Pl}_{>i}$ is the plausibility function corresponding to the combination of source triples $i+1, \ldots, k$. Since the first term does not depend on the choice of ω_i, $\mathrm{P}_{\mathrm{DS}}(\Omega'_{\omega_i})$ is proportional to $\mathrm{P}_i(\omega_i)\mathrm{Pl}_{>i}(X \cap \Gamma_i(\omega_i))$.

Unfortunately $\mathrm{Pl}_{>i}$ cannot be easily computed. If however, we can approximate it by a function Pl_i^* that we can efficiently compute, then we can pick ω_i with chance $(k_i)^{-1}\mathrm{P}_i(\omega_i)\mathrm{Pl}_i^*(X \cap \Gamma_i(\omega_i))$, and multiply the weight at that stage of the algorithm by $k_i/(\mathrm{Pl}_i^*(X \cap \Gamma_i(\omega_i)))$, where k_i is the appropriate normalisation constant.

[Moral and Salmerón, 99] suggests approximating this combined plausibility function by storing only a limited number of focal sets, using a method similar to that described above in 6.2. Although their experimental results suggest that this can perform well, their approximation of combined plausibility seems likely to become a poor one for larger problems (see the discussion above in 6.2), perhaps limiting the benefits of this new algorithm. However, there may be other approximations of plausibility that scale up better.

8 Computations for Decision-Making

Here we describe some functions which can be used in decision-making with Dempster-Shafer theory. First we describe lower and upper expected utility with respect to a mass/belief function, then Philippe Smets' pignistic probability is introduced. Finally we discuss the computation of these functions.

8.1 Lower and Upper Expected Utility

Let m be a mass function over Θ with associated belief function Bel; the associated set of compatible measures $\mathcal{P}$ is defined to be the set of probability measures π over Θ such that π dominates Bel, i.e., for all $A \subseteq \Theta$, $\pi(A) \geq \mathrm{Bel}(A)$ [Dempster, 67]. If π dominates Bel then Pl dominates π, where Pl is the plausibility function associated with m. Belief function Bel is the lower envelope of $\mathcal{P}$ and Pl is the upper envelope, i.e., for $A \subseteq \Theta$, $\mathrm{Bel}(A) = \inf\{\pi(A) : \pi \in \mathcal{P}\}$, and $\mathrm{Pl}(A) = \sup\{\pi(A) : \pi \in \mathcal{P}\}$.

Belief functions have been suggested as a representation of certain convex sets of probability functions, so that e.g., Bel is a representation of $\mathcal{P}$, see, for example, [Fagin and Halpern, 89; Wasserman, 90; Jaffray, 92]. Obviously, then $\mathcal{P}$ has a clear interpretation. In Dempster-Shafer theory, the connection between belief functions and sets of Bayesian probability functions is slightly more controversial (see e.g., [Shafer, 90]). However, Dempster intended that

the set of compatible probability functions be used in decision-making (see also [Dempster and Kong, 87]). Shafer, at least to some extent, goes along with this, in that he suggests in [Shafer, 81, page 22] that belief functions may be used for betting (and hence decision-making), by using lower expectation. (Although his approach does not explicitly deal with $\mathcal{P}$, it leads, as he points out, to the same values of upper and lower expected utility.) We briefly describe how this is done.

Suppose U is a function from Θ to $\mathbb{R}$, which for example could be a utility function. If our beliefs about Θ could be described by a Bayesian probability distribution $\pi : \Theta \to [\,0, 1\,]$ then we could calculate the value of expected utility $E_\pi[U] = \pi \cdot U = \sum_{\theta \in \Theta} \pi(\theta) U(\theta)$. If instead Bel summarises our beliefs about Θ, then it is natural to consider the lower expectation $E_*[U]$ and upper expectation $E^*[U]$ defined by $E_*[U] = \inf_{\pi \in \mathcal{P}} \pi \cdot U$ and $E^*[U] = \sup_{\pi \in \mathcal{P}} \pi \cdot U$. These can be expressed (see [Shafer, 81; Wilson, 93b]) in a computationally more convenient form: $E_*[U] = \sum_{A \subseteq \Theta} \mathrm{m}(A) U_*(A)$ and $E^*[U] = \sum_{A \subseteq \Theta} \mathrm{m}(A) U^*(A)$, where $U_*(A) = \min_{\theta \in A} U(\theta)$ and $U^*(A) = \max_{\theta \in A} U(\theta)$.

8.2 Pignistic Probability

An alternative approach to decision-making with belief functions is given by Philippe Smets in his variant of Dempster-Shafer theory, *the Transferable Belief Model*, see e.g., [Smets, 89; Smets and Kennes, 94]. This involves picking a particular element of the set of compatible measures $\mathcal{P}$, known as the *pignistic probability* , and calculating expected utility with respect to this.

The *pignistic probability measure* ρ associated with mass function m is defined by

$$\text{for } A \subseteq \Theta, \quad \rho(A) = \sum_{B \subseteq \Theta} \mathrm{m}(B) \frac{|B \cap A|}{|B|}.$$

ρ is thus the result of distributing each mass $\mathrm{m}(B)$ equally over the elements of B.

There are certainly advantages of generating a single probability measure to use for decision-making, as we're much more likely to get a uniquely preferred option; however it might be argued that picking any element of $\mathcal{P}$ over and above the others is unjustified. In particular, the choice of ρ has the problem that it is very much dependent on the choice of frame used to represent the problem (so if we refine Θ before generating the pignistic probabilities we get an essentially different result). An extreme example is when $\mathrm{m}(\Theta) = 1$ (the vacuous mass function [Shafer, 76]), when the pignistic probability distribution is the uniform distribution; the usual criticisms (see e.g., [Wilson, 92a]) of the notorious Principle of Indifference can then be used to argue against this. For more discussion of these issues see [Wilson, 93b].

8.3 Computation of the Functions

Lower and upper expected utility, pignistic probability, as well as belief, plausibility and commonality can all be expressed in the form $\sum \mathrm{m}(B)F(B)$ for different choices of the function $F : 2^\Theta \to \mathbb{R}$. For lower expected utility we set $F = U_*$, for upper expected utility we set $F = U^*$. For the pignistic probability of $A \subseteq \Theta$ we set $F(B) = \frac{|B \cap A|}{|B|}$; for $\mathrm{Bel}(A)$ we define $F(B)$ to be 1 iff $B \subseteq A$, and define it to be 0 otherwise; $\mathrm{Pl}(A)$ and $\mathrm{Q}(A)$ can be defined in a similar fashion.

This means that, if we have m represented in an ordered list or binary tree, we can compute these functions easily, since the expression equals $\sum_{B \in \mathcal{F}_\mathrm{m}} \mathrm{m}(B)F(B)$.

A more challenging situation is where m is the combination of a number of mass functions, and we're interested in e.g., lower expected utility. The combined mass function may have a very large number of focal sets so the exact methods of section 5 will often not be feasible. Using the source triples model (see sections 2.2 and 2.4.2) we can write $\sum_{B \subseteq \Theta} \mathrm{m}(B)F(B)$ as $\sum_{\omega \in \Omega} \mathrm{P_{DS}}(\omega)F(\Gamma(\omega))$, which is the expected value of the function $\Gamma \circ F$ with respect to $\mathrm{P_{DS}}$. The Monte-Carlo algorithms of 7.1, 7.2 and 7.3 all involve finding a way of picking an element of Ω with distribution $\mathrm{P_{DS}}$; we can use these methods to pick $\omega \in \Omega$, and then record the value $F(\Gamma(\omega))$; we repeat this many times and compute the average of the values of $F(\Gamma(\omega))$; this will then be our estimate of $\sum_{B \subseteq \Theta} \mathrm{m}(B)F(B)$. Adapting the Importance Sampling algorithms (7.4.1 and 7.4.2) is also straight-forward.

9 Exact Algorithms for Calculating Combined Belief over Product Sets

All the methods described above for calculating combined belief require the elements of Θ to be listed; if Θ is a product set formed from a sizeable number of frames then this will not be feasible, and so other methods must be used. In this section we describe and discuss the use of the local computation methods of Glenn Shafer and Prakash Shenoy (see e.g., [Shenoy and Shafer, 90]) for calculating combined belief in product spaces; a more general framework is described in [Shafer, Shenoy and Mellouli, 87] (see also [Kohlas and Monney, 95, chapter 8]).

For a fuller treatment of the subject of local computation and the literature, the reader should refer to the chapter by Jürg Kohlas and Prakash Shenoy in this volume. Papers which specifically deal with the computations in Dempster-Shafer theory on product sets include [Kong, 86; Shenoy and Shafer, 86; Shafer, Shenoy, and Mellouli, 87; Dempster, and Kong, 88; Xu, 91, 92; Xu and Kennes, 94; Bissig, Kohlas, and Lehmann, 97; Lehmann, and Haenni, 99].

9.1 Subsets of Product Sets

Let Ψ be finite non-empty set of variables. Associated with each variable $Y \in \Psi$ is its set of possible values (or frame of discernment) Θ_Y. For $s \subseteq \Psi$ define Θ_s to be the product set $\prod_{Y \in s} \Theta_Y$. Elements of Θ_s are called *configurations of* s, or *s-configurations*. Hence a configuration $\mathbf{x}$ of s may be considered as a function on s such that for $Y \in s$, $\mathbf{x}(Y) \in \Theta_Y$. For $r \subseteq s \subseteq \Psi$, let $\pi^s_r : \Theta_s \to \Theta_r$ be the natural projection function, defined as follows: for $\mathbf{x} \in \Theta_s$, $\pi^s_r(\mathbf{x})$ is just $\mathbf{x}$ restricted to r, i.e., for $Y \in r$, $\pi^s_r(\mathbf{x})(Y) = \mathbf{x}(Y)$. We will usually use a briefer notation: $\mathbf{x}^{\downarrow r}$ instead of $\pi^s_r(\mathbf{x})$. Function π^s_r is extended to subsets of Θ_s: for $A \subseteq \Theta_s$, $\pi^s_r(A)$ and $A^{\downarrow r}$ are both defined to be $\{\mathbf{x}^{\downarrow r} : \mathbf{x} \in A\}$. This operation is known as 'marginalising A to r'.

We will also be concerned, for $r \subseteq s \subseteq \Psi$, with $\rho^s_r = (\pi^s_r)^{-1}$, the set inverse of π^s_r, given by $\rho^s_r(\mathbf{y}) = \{\mathbf{x} \in \Theta_s : \mathbf{x}^{\downarrow r} = \mathbf{y}\}$, which will usually written as $\mathbf{y}^{\uparrow s}$. Again we extend to subsets: for $B \subseteq \Theta_r$, let $\rho^s_r(B)$, normally written as $B^{\uparrow s}$, $= \bigcup_{\mathbf{y} \in B} \mathbf{y}^{\uparrow s}$. We will refer to this operation as *vacuously extending B to s*.

The pair (Θ_r, π^s_r) is a coarsening of Θ_s and vacuous extension ρ^s_r is the associated refining (see section 3.4).

For example, let $\Theta_Y = \{a, b, c\}$ and $\Theta_Z = \{d, e\}$. Then $\Theta_{\{Y,Z\}} = \Theta_Y \times \Theta_Z = \{(a,d), (a,e), (b,d), (b,e), (c,d), (c,e)\}$. If $\mathbf{x} = (a, e)$ then $\mathbf{x}^{\downarrow\{Y\}} = a$. If $A = \{(a,e), (c,d), (c,e)\}$ then $A^{\downarrow\{Y\}} = \{a, c\}$. If $B = \{a, c\}$ then $B^{\uparrow\{Y,Z\}} = \{(a,d), (a,e), (c,d), (c,e)\}$.

For each variable Y it is assumed that there is precisely one correct (but usually unknown) value in Θ_Y. A configuration $\mathbf{x}$ of variables s is considered as a representation of the proposition that for each variable Y in s, the correct value of Y is $\mathbf{x}(Y)$. Hence there is precisely one true configuration of s, that consisting of all the correct values for all the variables in s. For $A \subseteq \Theta_s$, the set A of configurations of s is understood to represent the proposition that one of the configurations in A is the true one. With this interpretation one can see that, in the above example, the set $B = \{a, c\}$ represents the same proposition as $B^{\uparrow\{Y,Z\}} = \{(a,d), (a,e), (c,d), (c,e)\}$, since both tell us that the correct value variable Y is either a or c, but do not give us any (non-trivial) information about the correct value of variable Z. In general one can see that if $r \subseteq s \subseteq \Psi$, and $B \subseteq \Theta_r$, then B represents the same proposition (and so means the same thing) as $B^{\uparrow s}$; if we vacuously extend a set of configurations B we involve more variables but we do not assume anything about the values of those variables.

For $r \subseteq s \subseteq \Psi$ and $A \subseteq \Theta_s$, the set of r-configurations $A^{\downarrow r}$ gives us the same information as A gives about the correct values of variables in r (but of course tells us nothing about the variables in $s - r$).

Combination of sets of configurations: Suppose $s, t \subseteq \Psi$, $A \subseteq \Theta_s$ and $B \subseteq \Theta_t$. Define $A \otimes B$, the combination of A and B, to be $A^{\uparrow s\cup t} \cap B^{\uparrow s\cup t}$, which is a subset of $\Theta_{s\cup t}$.

With the interpretation given above, it can be seen that $A \otimes B$ means that the propositions represented by A and B are both true.

9.2 Mass Potentials on Product Sets

For set of variables $r \subseteq \Psi$, an *r-mass potential* is defined to be a mass potential over Θ_r. We can define r-mass functions, r-belief functions etc. analogously.

For $r \subseteq s \subseteq \Psi$, and r-mass potential m define s-mass potential $\mathrm{m}^{\uparrow s}$, called the vacuous extension of m to s, as follows: for $A \subseteq \Theta_s$, if there exists $B \subseteq \Theta_r$ with $B^{\uparrow s} = A$ then $\mathrm{m}^{\uparrow s}(A) = \mathrm{m}(B)$, otherwise $\mathrm{m}^{\uparrow s}(A) = 0$. Mass potential m is a mass function (i.e. $\mathrm{m}(\emptyset) = 0$) if and only if $\mathrm{m}^{\uparrow s}$ is a mass function. The set of focal sets of $\mathrm{m}^{\uparrow s}$ is just $\{B^{\uparrow s} : B \in \mathcal{F}_{\mathrm{m}}\}$, where $\mathcal{F}_{\mathrm{m}}$ is the set of focal sets of m. Since B and $B^{\uparrow s}$ represent the same proposition, m and $\mathrm{m}^{\uparrow s}$ can be considered as equivalent semantically.

Abbreviate the associated unnormalised commonality function $\mathrm{Q}_{\mathrm{m}^{\uparrow s}}$ to $\mathrm{Q}^{\uparrow s}$. For $A \subseteq \Theta_s$, we have $\mathrm{Q}^{\uparrow s}(A) = \mathrm{Q}(A^{\downarrow r})$ where Q is the unnormalised commonality function associated with m.

For $r \subseteq s \subseteq \Psi$ and s-mass potential m define $\mathrm{m}^{\downarrow r}$, known as the r-marginal of m, as follows: for $B \subseteq \Theta_r$ let $\mathrm{m}^{\downarrow r}(B) = \sum \{\mathrm{m}(A) : A \subseteq \Theta_s, A^{\downarrow r} = B\}$. $\mathrm{m}^{\downarrow r}$ is a mass function if and only if m is.

Define $\mathrm{Bel}^{\downarrow r}$ to be $\mathrm{Bel}_{\mathrm{m}^{\downarrow r}}$, i.e., the unnormalised belief function associated with $\mathrm{m}^{\downarrow r}$. The values of $\mathrm{Bel}^{\downarrow r}$ are determined by the equation: for $B \subseteq \Theta_r$, $\mathrm{Bel}^{\downarrow r}(B) = \mathrm{Bel}(B^{\uparrow s})$, where Bel is the unnormalised belief function associated with m.

Suppose $s, t \subseteq \Psi$, m is an s-mass potential, and n is a t-mass potential. If $s \neq t$ then $\mathrm{m} \oplus \mathrm{n}$ has not been defined since the mass potentials are over different frames. However m is semantically equivalent to $\mathrm{m}^{\uparrow s \cup t}$ and n is semantically equivalent to $\mathrm{n}^{\uparrow s \cup t}$, and these two are over the same frame. The combination of m and n, $\mathrm{m} \oplus \mathrm{n}$, is thus defined to be $\mathrm{m}^{\uparrow s \cup t} \oplus \mathrm{n}^{\uparrow s \cup t}$, where $\oplus$ on the right-hand side is the usual combination using Dempster's rule. Similarly we define the unnormalised combination $\mathrm{m} \otimes \mathrm{n}$ to be $\mathrm{m}^{\uparrow s \cup t} \otimes \mathrm{n}^{\uparrow s \cup t}$.

9.3 Product Set Propagation Algorithms

Suppose we are given, for $i = 1, \ldots, k$, s_i-mass function m_i, where $s_i \subseteq \Psi$; (the subsets s_i should be thought of as being fairly small—the methods described are not efficient otherwise). We are interested in the combined effect of this information, i.e., $\bigoplus_{i=1}^{k} \mathrm{m}_i$. We can leave the normalisation stage until the end, so let m be the mass potential $\mathrm{m} = \bigotimes_{i=1}^{k} \mathrm{m}_i$. This is a mass potential on the frame generated by variables $s_1 \cup \cdots \cup s_k$, which will typically be a huge product set, so we are certainly not usually going to be able to efficiently compute all focal sets of m explicitly. However, we may be particularly interested in the

impact on a set of variables,[13] say, s_1, so we will want to calculate $\mathrm{m}^{\downarrow s_1}$, and associated (normalised) belief values. Direct computation of this (by computing the combination m and then marginalising) will usually be infeasible. Prakash Shenoy and Glenn Shafer have shown how this can, in certain cases, be computed using combinations on much smaller frames, see e.g., [Shenoy and Shafer, 90].

There are two stages with their approach. The first stage takes as input the set of subsets $s_1, \ldots, s_k$ and returns another set of subsets $\mathcal{R}$ which is a hypertree (see below) and covers $s_1, \ldots, s_k$ (i.e., for each s_i there exists $r \in \mathcal{R}$ with $r \supseteq s_i$). The second stage uses the constructed hypertree cover to compute $\mathrm{m}^{\downarrow s_1}$ using a sequence of steps, each of which consists of a combination followed by a marginalisation, where the combination is on a frame Θ_u where $u \subseteq r$ for some $r \in \mathcal{R}$. This means that, if it is possible to find $\mathcal{R}$ such that each $r \in \mathcal{R}$ is small, then the combinations are all performed on frames of manageable size.

Kohlas and Shenoy (this volume) present the algorithm in a simpler way by considering deletion sequences, which avoids the need to talk about hypertrees. However, since we want to consider computational efficiency, which is closely linked to the hypertree cover, it makes sense to consider the latter explicitly.

9.3.1 Stage One: finding a hypertree cover

A set $\mathcal{R}$ of subsets of Ψ is said to be a hypertree [Shenoy and Shafer, 90] if it can be ordered as $r_1, \ldots, r_l$ where for each $2 \leq j \leq l$ there exists $j' \in \{1, \ldots, j-1\}$ such that $\emptyset \neq r_j \cap \bigcup_{i=1}^{j-1} r_i \subseteq r_{j'}$. The elements of a hypertree are called hyperedges. Such a sequence $r_1, \ldots, r_l$ of the hyperedges of $\mathcal{R}$ is said to be *a hypertree construction sequence for* $\mathcal{R}$.

Stage One involves finding a hypertree cover of $s_1, \ldots, s_k$, i.e., a hypertree $\mathcal{R}$ such that for each s_i there exists $r \in \mathcal{R}$ with $r \supseteq s_i$. We choose a hypertree construction sequence $r_1, \ldots, r_l$ of $\mathcal{R}$ such that $r_1 \supseteq s_1$.

It's worth spending some time trying to get a good hypertree cover as it can hugely affect the complexity of the computation. One measure[14] of 'badness' of a hypertree cover is the size of the largest product set associated with a hyperedge in $\mathcal{R}$, i.e., the maximum value of $\prod_{Y \in r_j} |\Theta_Y|$ as r_j ranges over the hyperedges in $\mathcal{R}$; the complexity of exact methods of computation can be exponentially related to this value. See Kohlas and Shenoy (this volume) section 3, for references for finding hypertree covers (since finding a good deletion sequence is essentially the same problem).

[13] We only consider here computation of one such marginal; however, there are efficient algorithms for computing all such marginal mass potentials, e.g., [Shenoy and Shafer, 90, section 3.4; Xu, 95; Bissig, Kohlas and Lehmann, 97; Kohlas and Shenoy, this volume, section 4.2].

[14] This measure is not always appropriate; for example, for some algorithms, more important is the sizes of product sets associated with intersections of hyperedges.

9.3.2 Stage Two: perform the sequence of combination-marginalisations

Recall $\mathrm{m} = \bigotimes_{i=1}^{k} \mathrm{m}_i$. We compute $\mathrm{m}^{\downarrow s_1}$ by computing $\mathrm{m}^{\downarrow r_1}$ and then marginalising to s_1. To calculate $\mathrm{m}^{\downarrow s_1}$ we first associate each m_i with a hyperedge r^{m_i} of our chosen hypertree cover $\mathcal{R}$ such that $r^{\mathrm{m}_i} \supseteq s_i$.

For $j = 1, \ldots, l$, let $\Psi_j = \bigcup_{i \leq j} r_i$. Let n be the unnormalised combination of all the mass potentials associated with r_l, i.e., $\bigotimes \{\mathrm{m}_i : r^{\mathrm{m}_i} = r_l\}$, and let $u \subseteq r_l$ be its associated set of variables[15]. We compute $\mathrm{n}' = \mathrm{n}^{\downarrow (u \cap \Psi_{l-1})}$. We also associate n' with a hyperedge r such that $r \supseteq u \cap \Psi_{l-1}$ (the definition of a hypertree makes this possible).

Using properties of mass potentials (see [Shenoy and Shafer, 90]) it can be shown that $\mathrm{m}^{\downarrow r_1} = (\bigotimes_{i=1}^{k} \mathrm{m}_i)^{\downarrow r_1}$ can be rewritten as $(\mathrm{n}' \otimes \bigotimes_{i \in \chi} \mathrm{m}_i)^{\downarrow r_1}$ where χ are the indices of the mass potentials not yet combined, i.e., $\chi = \{i : r^{\mathrm{m}_i} \neq r_l\}$.

We can then repeat the process, considering the combination of all mass potentials associated with r_{l-1}, letting u' be the set of variables associated with this combination, computing the marginalisation of the combination to $u' \cap \Psi_{l-2}$, and so on, until we have calculated $\mathrm{m}^{\downarrow r_1}$. Finally we can marginalise this to get $\mathrm{m}^{\downarrow s_1}$, normalise this, and we can then calculate the values of belief of subsets of Θ_{s_1} of interest.

Before giving two methods for performing the combination-marginalisations step needed in Stage 2 (9.3.4 and 9.3.5), we discuss a special case.

9.3.3 The special case of subsets

An important special case is when each of the input mass functions m_i has only a single focal set: i.e., for $i = 1, \ldots, k$, there exists C_i with $\mathrm{m}_i(C_i) = 1$. Combination and marginalisation both preserve this property, and $\mathrm{m}_i^{\downarrow t} = C_i^{\downarrow t}$ and $\mathrm{m}_i \otimes \mathrm{m}_j = C_i \otimes C_j$ where the operations on subsets are those defined in section 9.1.

As described above, the basic step in the method involves combination of a number of mass potentials $\mathrm{m}_i : i \in \sigma$ associated with a hyperedge r, marginalised to a subset t of another hyperedge, which can be done by computing $\left(\bigcap_{i \in \sigma} C_i^{\uparrow r}\right)^{\downarrow t}$. An obvious algorithm for doing this takes time proportional to $|\Theta_r| \times |\sigma|$. Unless the number of mass potentials we're combining is huge, the term to be concerned about is $|\Theta_r| = \prod_{Y \in r} |\Theta_Y|$.

The complexity of the computation for this case is hence linearly related to the size of the largest product set associated with a hyperedge in the hypertree cover used.

[15] Normally u will be equal to r_l; this always happens, for example, if the hypertree is generated by a deletion sequence.

9.3.4 Fast Möbius Transform method for Stage Two

The idea behind this method is to perform combinations with (unnormalised) commonality functions, and marginalisations with (unnormalised) belief functions, because of the simple forms of these operations for those functions. Hence we convert to commonality functions before combination, use these to combine, and convert the combination to the belief function representation; this is then marginalised.

The basic step in the method involves combination of a number of mass potentials $\mathrm{m}_i : i \in \sigma$ associated with a hyperedge r, marginalised to a subset t of another hyperedge, i.e., computing $\left(\bigotimes_{i\in\sigma} \mathrm{m}_i^{\uparrow u}\right)^{\downarrow t}$, where set of variables u equals $\bigcup_{i\in\sigma} s_i$ $(\subseteq r)$.

Since we will be using associated unnormalised belief and commonality functions to represent each m_i and the resulting mass potential, we must (i) convert our representation for each m_i to the associated unnormalised commonality function Q_i. Then (ii) we can use the simple form of Dempster's rule for commonality functions to compute the combined unnormalised commonality function. Finally (iii) we convert this to its associated unnormalised belief function, from where the values of the marginalised belief function can be read off. In more detail:

(i) There are two cases:

  (a) if m_i is one of the input mass functions, then we use the Fast Möbius Transform (ii) (see section 4.2) to convert m_i to the associated unnormalised commonality function Q_i.

  (b) if m_i is the result of a previous computation then our representation for it is as its associated unnormalised belief function Bel_i. We then use the appropriate Fast Möbius Transform (iii) (see section 4.2) to convert to the associated unnormalised commonality function Q_i.

(ii) Let Q be the combined unnormalised commonality function, i.e., that associated with $\bigotimes_{i\in\sigma} \mathrm{m}_i^{\uparrow u}$. For $A \subseteq \Theta_u$, $\mathrm{Q}(A) = \prod_{i\in\sigma} \mathrm{Q}_i^{\uparrow u}(A)$ which can be computed as $\prod_{i\in\sigma} \mathrm{Q}_i(A^{\downarrow s_i})$.

(iii) Q can then be converted to the associated unnormalised belief function Bel. The values of $\mathrm{Bel}^{\downarrow t}$ are then given by $\mathrm{Bel}^{\downarrow t}(B) = \mathrm{Bel}(B^{\uparrow u})$ for $B \subseteq \Theta_t$.

One can get a good idea of the complexity by considering stage (iii). This involves $|\Theta_u|2^{|\Theta_u|-1}$ additions, where $|\Theta_u| = \prod_{Y\in u} |\Theta_Y|$. Note that this is true for all cases, not just the worst case. Even for a set u consisting of 5 boolean variables, $|\Theta_u| = 32$, so the number of additions is about 70 billion. Problems in which there's a hyperedge (i.e., a set u) consisting of 6 boolean variables, or

4 three-valued variables, or 3 four-valued variables are all well beyond the reach of today's computers.

This double exponentiality means that the approach is only feasible for a very restricted class of problems, where all the product sets associated with hyperedges are small.

A more direct approach to moving between m and Q is suggested in [Bissig, Kohlas and Lehmann, 97]. If one has a list of the focal sets with corresponding masses (e.g., as an ordered list, see 3.1.2), the equation $\mathrm{Q}(A) = \sum_{B \supseteq A} \mathrm{m}(B)$, can be used to compute the values of Q on $\mathcal{F}$. Conversely, suppose that one has a list of values of Q on some set $\mathcal{G}$ known to contain all the focal sets $\mathcal{F}$. If one orders $\mathcal{G}$ as $A_1, \ldots, A_{|\mathcal{G}|}$ in any way such that $A_i \supseteq A_j$ implies $i \leq j$ then m can be recovered by applying the equation $\mathrm{m}(A) = \mathrm{Q}(A) - \sum_{B \supset A} \mathrm{m}(B)$ (where $B \supset A$ means 'B is a proper superset of A') sequentially to $A_1, A_2, \ldots$. Very similar methods can be used to move between m and Bel. These computations are, at worst, quadratic in $|\mathcal{F}|$ (or $|\mathcal{G}|$), so if the number of focal sets is very much smaller than $2^{|\Theta|}$ this approach can be much faster than using the Fast Möbius Transform. The computation of the combined commonality function (see (ii) above) can be done iteratively (to allow more efficient computation of the set of focal sets of the combination), similarly to the combination of several mass potentials in 5.1.2 and 9.3.5. However the final conversion stage (see (iii) above) to m and/or Bel may make this approach often worse than the mass-based approach below.

9.3.5 Mass-based algorithm for Stage Two

An alternative method is where we only deal with mass potentials, using the method of section 5.1 for the combination; the marginalisation step can then be performed using the equation $\mathrm{m}^{\downarrow t}(B) = \sum \{\mathrm{m}(A) : A \subseteq \Theta_u, A^{\downarrow t} = B\}$, for $B \subseteq \Theta_t$.

This approach will tend to suffer from the same double exponentiality problems as the Fast Möbius Transform method (9.3.4), and can be even worse than that method for some problems. However, there are many situations where the mass-based approach will be much faster.

For example, suppose we want to combine an $\{Y_1, Y_2, Y_3\}$-mass potential with an $\{Y_1, Y_4, Y_5\}$-mass potential and then marginalise to $\{Y_2, Y_3, Y_4\}$, where all the variables are boolean. Each mass potential has at most $2^{2^3} = 256$ focal sets so an upper bound for the number of multiplications needed for the combination is $256^2 = 2^{16}$, and none are needed for the marginalisation. The number of additions needed for combination and marginalisation is less than this. The combination also needs at most 2^{16} intersections, each of which requires at most 2^5 very fast operations (one-digit binary multiplications); the dominant term will thus probably be the 2^{16} multiplications. The Fast Möbius Transform method, on the other hand, requires 2^{32} multiplications, making it

much slower.

A more extreme example is if we suppose that, as well as the $\{Y_1, Y_2, Y_3\}$-mass potential and the $\{Y_1, Y_4, Y_5\}$-mass potential, we have to combine a $\{Y_1, Y_2, Y_3, Y_4, Y_5, Y_6\}$-mass potential, which is one of the inputs of the problem, and only had 4 focal sets (the input mass functions will very often have small numbers of focal sets). An upper bound for the number of multiplications needed by the direct mass-based approach is $2^{16} \times 4 = 2^{18}$. However the Fast Möbius Transform method would need 2^{65} multiplications which is completely unfeasible.

However, after a few stages of combination and marginalisation, the number of focal sets will tend to get very large, so all these methods have severe computational problems.

9.3.6 Conditioning by marginalised combined core

Earlier, it was pointed out that initially conditioning the input mass functions by their combined core could in some cases greatly improve the efficiency of frame-based algorithms. A similar operation can be applied in the case of product set frames of discernment.

Let $\mathcal{C}_i \subseteq \Theta_{s_i}$ be the core of mass function m_i. With product sets one cannot generally efficiently calculate (explicitly) the combined core $\bigotimes_{i=1}^{k} \mathcal{C}_i$ as it's a subset of an often huge product set Θ_Ψ. However, one can efficiently compute for each i, $\mathrm{m}'_i = \mathrm{m}_i \otimes (\bigotimes_{i=1}^{k} \mathcal{C}_i)^{\downarrow s_i}$, since combination of subsets is fast (see 9.3.3). It can be shown that the combination $\bigotimes \mathrm{m}'_i$ is the same as $\bigotimes \mathrm{m}_i$, so we can replace each m_i by m'_i, without changing the combination. The main reason for doing this is that it can happen that m'_i has many fewer focal sets than m_i.

10 Monte-Carlo Methods for Product Sets

In this section, various Monte-Carlo methods from section 7 are applied to the case when the frame of discernment is a product set. All the algorithms can benefit from first conditioning all the input mass functions by the marginalised combined core (see section 9.3.6).

It is of course possible to apply the algorithms of section 7 directly, ignoring the structure of Θ_Ψ and just treating it like any other frame. For example, with the commonality-based importance sampling algorithm (section 7.4.1), the time required will be roughly proportional to $|\Theta_\Psi|$, which may be feasible for not too large problems; it will, for example, be more efficient than the exact product set propagation method using the Fast Möbius Transform (section 9.3.4) if $|\Theta_\Psi|$ is very much smaller than 2^N where $N = \max\{|\Theta_r| : \text{ hyperedges } r\}$.

However, very often, the product set will be too large to take this approach, so we consider methods that use the structure of the product set.

10.1 Source Triples for Product Sets

As before, it's convenient to use the source triples representation for Monte-Carlo algorithms; each m_i is represented by an equivalent source triple $(\Omega_i, \mathrm{P}_i, \Gamma_i)$ over Θ_{s_i}, so that for all $B \subseteq \Theta_{s_i}$, $\mathrm{m}_i(B) = \sum \{\mathrm{P}_i(\omega_i) : \omega_i \in \Omega_i, \Gamma_i(\omega_i) = B\}$. We have to slightly amend the definition (section 2.4.2) of the combined source triple $(\Omega, \mathrm{P}_{\mathrm{DS}}, \Gamma)$: define $\Gamma' \colon \Omega^{\times} \to 2^{\Theta}$ by $\Gamma'(\omega) = \bigotimes_{i=1}^{k} \Gamma_i(\omega(i))$, and as before, let Γ be Γ' restricted to Ω; the other parts of the definition are unchanged.

To slightly simplify the presentation we'll assume, without any real loss of generality, that $\bigcup_{i=1}^{k} s_i = \Psi$. As discussed above, $A \subseteq \Theta_{s_i}$ is semantically equivalent to $A^{\uparrow\Psi} \subseteq \Theta_{\Psi}$; each $\Gamma_i(\omega(i))$ is semantically equivalent to $\Gamma_i(\omega(i))^{\uparrow\Psi}$, and so $\bigotimes_{i=1}^{k} \Gamma_i(\omega(i))$ is semantically equivalent to $\bigcap_{i=1}^{k} (\Gamma_i(\omega(i))^{\uparrow\Psi}$. Hence this definition is consistent with the earlier definition of the combined source triple.

10.2 Simple Monte-Carlo Method

We'll assume that any set A whose belief we want to find, is a subset of Θ_t for some hyperedge t in the hypertree cover. If we want to find the belief of any other set s_0, then we choose a hypertree cover of $s_0, s_1, \ldots, s_k$ including s_0.

The simple Monte-Carlo algorithm (see section 7.1) can be applied easily to the product set case:

For each trial:
  for $i = 1, \ldots, k$
    pick $\omega_i \in \Omega_i$ with chance $\mathrm{P}_i(\omega_i)$
  let $\omega = (\omega_1, \ldots, \omega_k)$
  if $(\Gamma(\omega))^{\downarrow t} = \emptyset$
    then restart trial
    else if $(\Gamma(\omega))^{\downarrow t} \subseteq A$
      then trial succeeds
      else trial fails

Checking conditions $(\Gamma(\omega))^{\downarrow t} = \emptyset$ and $(\Gamma(\omega))^{\downarrow t} \subseteq A$, i.e., $\left(\bigotimes_{i=1}^{k} \Gamma_i(\omega(i))\right)^{\downarrow t} \subseteq A$, can be done using the algorithm for propagating subsets (see 9.3.3).

Since $A \subseteq \Theta_t$ can be viewed as a representation of $A^{\uparrow\Psi}$, and the condition $(\Gamma(\omega))^{\downarrow t} \subseteq A$ is equivalent to $(\Gamma(\omega)) \subseteq A^{\uparrow\Psi}$, this can be seen to essentially just be a rewriting for product sets of the frame-based algorithm.

Again there is the problem that if the conflict between the input mass functions is very high (i.e., $\mathrm{P}'(\Omega)$ is very small), then the algorithm will be slow. One idea to improve this situation is suggested by the observation that the conflict between a pair of mass functions m_i and m_j will tend to be less if the intersection $s_i \cap s_j$ of their associated sets of variables is small; (an extreme case is when $s_i \cap s_j = \emptyset$: then there can be no conflict between m_i and m_j). Therefore it may well be the case that most of the conflict between the input mass

functions/source triples arises from conflict between mass functions associated with the same hyperedge. We can remove this source of conflict by amending the algorithm, a single trial of which is described below:

For each hyperedge r in the hypertree cover $\mathcal{R}$ we consider the set M_r of associated mass functions, i.e., those with $r^{\mathrm{m}_i} = r$. We apply the random algorithm using commonality (see section 7.3) to pick a random focal set A_r of the combination $\bigotimes M_r$. The time needed for this will be roughly linearly related to $|\Theta_r|$ (ignoring the dependence on k, and with a fairly high constant factor), and is not affected by a high degree of conflict.

We then test to see if $\bigotimes_{r\in\mathcal{R}} A_r$ is empty. If it is, then we have to restart the trial. If it is non-empty we, as usual, test to see if $(\bigotimes_{r\in\mathcal{R}} A_r)^{\downarrow t} \subseteq A$.

Note that it may well be worth choosing the hypertree cover $\mathcal{R}$ so that the intersections between hyperedges r are small, as this can reduce the conflict, even if it means making some of the hyperedges larger.

10.3 Markov Chain Method

The adaption of the Markov Chain method of section 7.2 to the product set case is also straight-forward. The implementation of *operation*$_i$ involves finding which elements of Ω_i are consistent with the other co-ordinates of the current state ω^c, i.e., for which elements $\omega_i \in \Omega_i$ is $\Gamma_i(\omega_i) \otimes \bigotimes_{j\neq i} \Gamma_j(\omega_j^c)$ non-empty; this can be efficiently determined using the propagation of subsets (section 9.3.3) by checking the equivalent condition: $\Gamma_i(\omega_i) \otimes (\bigotimes_{j\neq i} \Gamma_j(\omega_j^c))^{\downarrow r^{\mathrm{m}_i}} \neq \emptyset$, where $r^{\mathrm{m}_i} \in \mathcal{R}$ is the hyperedge associated with m_i.

This algorithm suffers from the same problem as the frame-based algorithm (see section 7.2): it will not work (well) if Ω is unconnected (poorly connected). Also, for the product set case there seems to be no easy way of testing connectivity. However many connectivity problems will be internal to hyperedges, and these can be solved using a form of *blocking*: we apply the Markov Chain Algorithm of section 7.2 to the set of mass potentials $\{\mathrm{m}^r : r \in \mathcal{R}\}$, where m^r is the combination $\bigotimes_{\mathrm{m}\in M_r} \mathrm{m}$. We don't need to explicitly calculate the combinations $\bigotimes_{\mathrm{m}\in M_r} \mathrm{m}$, but, for a given r, we can use the random algorithm using commonality (section 7.3) to pick a new random focal set of m^r which is consistent with the other co-ordinates of the current state.

10.4 Importance Sampling Based on Consistency

This algorithm can also be easily adapted to product sets, in a similar fashion to the Markov Chain algorithm. As with the frame-based version of the algorithm (section 7.4.2), it is essential that each focal set of each input mass function is consistent with the combined core. This can be achieved by initially conditioning each input mass function by the marginalised combined core (see section 9.3.6).

The algorithm involves determining the set Δ_i^ω of consistent choices for the ith co-ordinate, i.e., $\{\omega_i \in \Omega_i : \bigotimes_{j=1}^{i} \Gamma_j(\omega_j) \neq \emptyset\}$, i.e., ω_i such that $\Gamma_i(\omega_i) \cap B^i$ is non-empty, where $B^i = (\bigotimes_{j=1}^{i-1} \Gamma_j(\omega_j))^{\downarrow s_i}$, which can be calculated using the method of propagating subsets in section 9.3.3.

This algorithm may work well, as there doesn't seem to be an obvious connection between the size of the frame of discernment and the variance of the estimate of belief; however, further analysis and experimental testing is needed to ascertain this.

11 Summary

The main problem to which this chapter is addressed is that of calculating, from a number of mass functions, values of Dempster-Shafer belief corresponding to their combination (using Dempster's rule). The following are the general conclusions of the chapter.

- For very small frames of discernment Θ, exact algorithms can be used to compute combined Dempster-Shafer belief. Approximation algorithms can probably also sometimes be useful when Θ is small.
- Monte-Carlo algorithms should be reasonably efficient for calculating combined belief, if Θ is not huge, with the best of the algorithms being roughly linear in $|\Theta|$.
- Where Θ is a huge product set, the local computation approach of Shafer and Shenoy can sometimes be applied, but exact algorithms appear to be practical only for very sparse situations.
- Several of the Monte-Carlo algorithms can be applied to this case, and seem to be much more promising; however it is still unclear which of these algorithms (if any) will work well in practice. Further development of Monte-Carlo algorithms for product sets seems to be the most important area for future research.

Acknowledgements

I'm grateful to Lluís Godo, Jürg Kohlas and Philippe Smets for their detailed and helpful comments.

References

Barnett, J.A., 81, Computational methods for a mathematical theory of evidence, in: *Proceedings IJCAI-81*, Vancouver, BC 868-875.

Bauer, M., 97, Approximation Algorithms and Decision-Making in the Dempster-Shafer Theory of Evidence—An Empirical Study, *International Journal of Approximate Reasoning* 17: 217–237.

Bissig, R., Kohlas, J., and Lehmann, N., 97, Fast-Division Architecture for Dempster-Shafer Belief Functions, *Proc. ECSQARU–FAPR'97*, D. Gabbay, R. Kruse, A. Nonnengart, and H. J. Ohlbach (eds.), Springer-Verlag, 198–209.

Dempster, A. P., 67, Upper and Lower Probabilities Induced by a Multi-valued Mapping. *Annals of Mathematical Statistics* 38: 325–39.

Dempster, A. P., 68, A Generalisation of Bayesian Inference (with discussion), *J. Royal Statistical Society* ser. B 30: 205–247.

Dempster, A. P., and Kong, A., 87, in discussion of G. Shafer, Probability Judgment in Artificial Intelligence and Expert Systems (with discussion) *Statistical Science*, 2, No.1, 3-44.

Dempster, A. P., and Kong, A., 88, Uncertain Evidence and Artificial Analysis, *Journal of Statistical Planning and Inference* 20 (1988) 355–368; also *Readings in Uncertain Reasoning*, G. Shafer and J. Pearl (eds.), Morgan Kaufmann, San Mateo, California, 1990, 522–528.

Dubois, D. and Prade, H., 90, Consonant Approximations of Belief Functions, *International Journal of Approximate Reasoning* 4: 419–449.

Fagin R., and Halpern, J. Y., 89, Uncertainty, Belief and Probability, *Proc., International Joint Conference on AI* (IJCAI-89), 1161-1167.

Feller, W., 50, *An Introduction to Probability Theory and Its Applications*, second edition, John Wiley and Sons, New York, London.

Gordon, J. and Shortliffe, E.H., 85, A Method of Managing Evidential Reasoning in a Hierarchical Hypothesis Space, *Artificial Intelligence* 26, 323-357.

Hájek, P., 92, Deriving Dempster's Rule, *Proc. IPMU'92*, Univ. de Iles Baleares, Mallorca, Spain, 73–75.

IJAR, 92, Special Issue of *Int. J. Approximate Reasoning*, 6:3, May 1992.

Jaffray, J-Y, 92, Bayesian Updating and Belief Functions, *IEEE Trans. SMC*, 22: 1144–1152.

Joshi, A.V., Sahasrabudhe, S. C., and Shankar, K., 95, Bayesian Approximation and Invariance of Bayesian Belief Functions, 251–258 of Froidevaux, C., and Kohlas, J., (eds.), *Proc. ECSQARU '95*, Springer Lecture Notes in Artificial Intelligence 946.

Kämpke, T., 88, About Assessing and Evaluating Uncertain Inferences Within the Theory of Evidence, *Decision Support Systems* 4, 433-439.

Kennes, R., and Smets, P., 90a, Computational Aspects of the Möbius transform, *Proc. 6th Conference on Uncertainty in Artificial Intelligence*, P. Bonissone, and M. Henrion, (eds.), MIT, Cambridge, Mass., USA, 344–351.

Kennes, R.,, and Smets, P., 90b, *Proceedings of IPMU Conference,* Paris, France, 99–101. Full paper in: Bouchon-Meunier, B., Yager, R. R., Zadeh, L. A., (eds.), *Uncertainty in Knowledge Bases,*, (1991), 14–23.

Kohlas, J., and Monney, P.-A., 95, *A Mathematical Theory of Hints*, Springer Lecture Notes in Economics and Mathematical Systems 425.

Kong, A., 86, *Multivariate belief functions and graphical models*, PhD dissertation, Dept. Statistics, Harvard University, USA.

Kreinovich, V., Bernat, A., Borrett, W., Mariscal, Y., and Villa, E., 92, Monte-Carlo Methods Make Dempster-Shafer Formalism Feasible, 175–191 of Yager, R., Kacprzyk, J., and Fedrizzi, M., (eds.), *Advances in the Dempster-Shafer Theory of Evidence*, John Wiley and Sons.

Laskey, K. B. and Lehner, P. E., 89, Assumptions, Beliefs and Probabilities, *Artificial Intelligence* 41 (1989/90): 65–77.

Lehmann, N., and Haenni, R., 99, An Alternative to Outward Propagation for Dempster-Shafer Belief Functions, *Proc. ECSQARU'99*, London, UK, July 99, Lecture Notes in Artificial Intelligence 1638, A. Hunter and S. Parsons (eds.), 256–267.

Levi, I., 83, Consonance, Dissonance and Evidentiary Mechanisms, in *Evidentiary value: Philosophical, Judicial and Psychological Aspects of a Theory*, P. Gärdenfors, B. Hansson and N. E. Sahlin, (eds.) C. W. K. Gleerups, Lund, Sweden.

Moral, S., and Salmerón, A., 99, A Monte Carlo Algorithm for Combining Dempster-Shafer Belief Based on Approximate Pre-computation, *Proc. ECSQARU'99*, London, UK, July 99, Lecture Notes in Artificial Intelligence 1638, A. Hunter and S. Parsons (eds.), 305–315.

Moral, S., and Wilson, N., 94, Markov Chain Monte-Carlo Algorithms for the Calculation of Dempster-Shafer Belief, *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94*, Seattle, USA, July 31–August 4, 1994, 269–274.

Moral, S., and Wilson, N., 96, Importance Sampling Monte-Carlo Algorithms for the Calculation of Dempster-Shafer Belief, *Proceedings of IPMU'96*, Vol. III, 1337-1344.

Orponen, P., 90, Dempster's rule is # P-complete, *Artificial Intelligence*, 44: 245–253.

Pearl, J., 88, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc. 1988, Chapter 9, in particular 455-457.

Pearl, J., 90a, Bayesian and Belief-Function Formalisms for Evidential Reasoning: a Conceptual Analysis, *Readings in Uncertain Reasoning*, G. Shafer and J. Pearl (eds.), Morgan Kaufmann, San Mateo, California, 1990, 540-574.

Pearl, J., 90b, Reasoning with Belief Functions: An Analysis of Compatibility, *International Journal of Approximate Reasoning*, 4(5/6), 363–390.

Provan, G., 90, A Logic-based Analysis of Dempster-Shafer Theory, *International Journal of Approximate Reasoning*, 4, 451-495.

Shafer, G., 76, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ.

Shafer, G., 81, Constructive Probability, *Synthese*, 48: 1–60.

Shafer, G., 82a, Lindley's paradox (with discussion), *Journal of the American Statistical Association* 7, No. 378, 325-351.

Shafer, G., 82b, Belief Functions and Parametric Models (with discussion), *J. Royal Statistical Society* ser. B, 44, No. 3, 322-352.

Shafer, G., 84, Belief Functions and Possibility Measures, Working Paper no.163, School of Business, The University of Kansas, Lawrence, KS, 66045, USA.

Shafer, G., 90, Perspectives on the Theory and Practice of Belief Functions, *International Journal of Approximate Reasoning* 4: 323-362.

Shafer, G., 92, Rejoinders to Comments on "Perspectives on the Theory and Practice of Belief Functions", *International Journal of Approximate Reasoning*, 6, No. 3, 445–480.

Shafer, G. and Logan, R., 87, Implementing Dempster's Rule for Hierarchical Evidence, *Artificial Intelligence* 33: 271-298.

Shafer, G., Shenoy, P. P., and Mellouli, K., 87, Propagating Belief Functions in Qualitative Markov Trees, *International Journal of Approximate Reasoning* 1: 349–400.

Shafer, G., and Tversky, A., 85, Languages and Designs for Probability Judgement *Cognitive Science* 9: 309–339.

Shenoy, P. P. and Shafer, G., 86, Propagating Belief Functions with Local Computations, *IEEE Expert*, 1 No.3: 43–52.

Shenoy, P. P., and Shafer, G., 90, Axioms for Probability and Belief Function Propagation, in *Uncertainty in Artificial Intelligence 4*, R. Shachter, T. Levitt, L. Kanal, J. Lemmer (eds.), North Holland, also in *Readings in Uncertain Reasoning*, G. Shafer and J. Pearl (eds.), Morgan Kaufmann, San Mateo, California, 1990, 575–610.

Smets, P., 88, Belief Functions, in *Non-standard Logics for Automated Reasoning*, P. Smets, E. Mamdami, D. Dubois and H. Prade (eds.), Academic Press, London.

Smets, P. 89, Constructing the Pignistic Probability Function in a Context of Uncertainty, in *Proc. 5th Conference on Uncertainty in Artificial Intelligence*, Windsor.

Smets, P., 90, The Combination of Evidence in the Transferable Belief Model, *IEEE Trans. Pattern Analysis and Machine Intelligence* 12, 447–458.

Smets, P., and Kennes, R., 94, The Transferable Belief Model, *Artificial Intelligence* 66:191–234.

Tessem, B., 93, Approximations for Efficient Computation in the Theory of Evidence, *Artificial Intelligence*, 61, 315–329.

Thoma, H.M., 89, *Factorization of Belief Functions*, Ph.D. Thesis, Department of Statistics, Harvard University, Cambridge, MA, USA.

Thoma, H.M., 91, Belief Function Computations, 269–308 in Goodman, I.R., Gupta, M.M., Nguyen, H.T., Roger, G.S., (eds.), *Conditional Logic in Expert Systems*, New York: North-Holland.

Voorbraak, F., 89, A Computationally Efficient Approximation of Dempster-Shafer Theory, *Int. J. Man-Machine Studies*, 30, 525–536.

Voorbraak, F., 91, On the justification of Dempster's rule of combination, *Artificial Intelligence* 48: 171–197.

Walley, P., 91, *Statistical Reasoning with Imprecise Probabilities*, Chapman and Hall, London.

Wasserman, L. A., 90, Prior Envelopes Based on Belief Functions, *Annals of Statistics* 18, No.1: 454-464.

Wilson, N., 87, On Increasing the Computational Efficiency of the Dempster-Shafer theory, Research Report no. 11, Sept. 1987, Dept. of Computing and Mathematical Sciences, Oxford Polytechnic.

Wilson, N., 89, Justification, Computational Efficiency and Generalisation of the Dempster-Shafer Theory, Research Report no. 15, June 1989, Dept. of Computing and Mathematical Sciences, Oxford Polytechnic.

Wilson, N., 91, A Monte-Carlo Algorithm for Dempster-Shafer Belief, *Proc. 7th Conference on Uncertainty in Artificial Intelligence*, B. D'Ambrosio, P. Smets and P. Bonissone (eds.), Morgan Kaufmann, 414-417.

Wilson, N., 92a, How Much Do You Believe?, *International Journal of Approximate Reasoning*, 6, No. 3, 345-366.

Wilson, N., 92b, The Combination of Belief: When and How Fast, *International Journal of Approximate Reasoning*, 6, No. 3, 377–388.

Wilson, N., 92c, Some Theoretical Aspects of the Dempster-Shafer Theory, PhD thesis, Oxford Polytechnic, May 1992.

Wilson, N., 93a, The Assumptions Behind Dempster's Rule, *Proceedings of the Ninth Conference of Uncertainty in Artificial Intelligence (UAI93)*, David Heckerman and Abe Mamdani (eds.), Morgan Kaufmann Publishers, San Mateo, California, 527–534.

Wilson, N., 93b, Decision-Making with Belief Functions and Pignistic Probabilities, *2nd European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU-93)*, Kruse, R., Clarke, M. and Moral, S., (eds.), Springer Verlag, 364–371.

Wilson, N., and Moral, S., 96, Fast Markov Chain Algorithms for Calculating Dempster-Shafer Belief, 672–676, *Proceedings of the 12th European Conference on Artificial Intelligence, (ECAI-96)*, Wahlster, W., (ed.), John Wiley and Sons.

Xu, H., 91, An efficient implementation of the belief function propagation, *Proc. 7th Conference on Uncertainty in Artificial Intelligence*, B. D'Ambrosio, P. Smets and P. Bonissone (eds.), Morgan Kaufmann, 425–432.

Xu, H., 92, An efficient tool for reasoning with belief functions, in Bouchon-Meunier, B., Valverde, L. and Yager, R.R., (eds.) *Uncertainty in intelligent systems*, North Holland, Elsevier Science, 215–224.

Xu, H., 95, Computing Marginals for Arbitrary Subsets from the Marginal Representation in Markov Trees, *Artificial Intelligence* 74:177-189.

Xu, H., and Kennes, R., 94, Steps towards an Efficient Implementation of Dempster-Shafer Theory, in Fedrizzi M., Kacprzyk J., and Yager R. R., (eds.) *Advances in the Dempster-Shafer Theory of Evidence*, John Wiley & Sons, Inc., 153–174.

Zadeh, L. A., 84. A Mathematical Theory of Evidence (book review) *The AI Magazine* 5 No. 3: 81-83.