Markov Chain Monte-Carlo Algorithms for the Calculation of Dempster-Shafer Belief

Serafin Moral
Departamento de Ciencias de la Computación e I. A.,
Universidad de Granada
18071 - Granada - Spain
smc@robinson.ugr.es

Nic Wilson
Department of Computer Science
Queen Mary and Westfield College
Mile End Rd., London E1 4NS, UK
nic@dcs.qmw.uk.ac

Abstract
A simple Monte-Carlo algorithm can be used to calculate Dempster-Shafer belief very efficiently unless the conflict between the evidences is very high. This paper introduces and explores Markov Chain Monte-Carlo algorithms for calculating Dempster-Shafer belief that can also work well when the conflict is high.

1. Introduction
Dempster-Shafer theory (Shafer 1976, 1990, Dempster 1967) is a promising method for reasoning with uncertain information. The theory involves splitting the uncertain evidence into independent pieces and calculating the combined effect using Dempster’s rule of combination. A major problem with this is the computational complexity of Dempster’s rule. The straightforward application of the rule is exponential (where the problem parameters are the size of the frame of discernment and the number of evidences). A number of methods have been developed for improving the efficiency, e.g., (Laskey & Lehner 1989, Wilson 1989, Provan 1990, Kennes & Smets 1990) but they are limited by the #P-completeness of Dempster’s rule (Orponen 1990). However, the precise value of Dempster-Shafer belief is of no great importance—it is sufficient to find an approximate value, within a small range of the correct value. Dempster’s formulation suggests a simple Monte-Carlo algorithm for calculating DS-belief, (Pearl 1988, Kámpke 1988, Wilson 1989, 1991). This algorithm, described in section 3, involves a large number of independent trials, each taking the value 0 or 1 and having an expected value of the Dempster-Shafer belief. This belief is then approximated as the average of the values of these trials. The algorithm can be used to efficiently calculate Dempster-Shafer belief unless the conflict between the evidences is very high. Unfortunately there are cases where the conflict will be very high (Shafer 1992) making this algorithm unusable for those cases.

Similar problems have been found for Monte-Carlo algorithms in Statistics, and also in Bayesian networks. A common solution is to use Markov Chain Monte-Carlo algorithms (Smith & Roberts 1993, Geyer 1992, Hrycej 1990) for which the trials are not independent, but are instead governed by a Markov Chain (Feller 1950). Such methods are used when it is very difficult to simulate independent realizations of some complicated probability distribution.

In this paper we develop Markov Chain algorithms for the calculation of Dempster-Shafer belief. Section 4 describes the algorithms and gives the convergence results. Convergence of the algorithms is dependent on a particular connectivity condition; a way of testing for this condition is given in section 5. Section 6 discusses the results of computer testing of the algorithm. Section 7 shows how the algorithm can be extended and applied to the calculation of Dempster-Shafer belief on logics, and on infinite frames, and section 8 briefly discusses some extensions to these algorithms which may work when the connectivity condition is not satisfied.

2. Belief Functions and Source Triples
Let be the finite set. is intended to represent a set of mutually exclusive and exhaustive propositions. A mass function over is a function such that and . Function is said to be a belief function over if there exists a mass function over with, for all , Clearly, to every mass function over there corresponds (with the above relationship) a unique belief function; conversely for every belief function over there corresponds a unique mass function (Shafer 1976).

Belief functions are intended as representations of subjective degrees of belief, as described in (Shafer...
A source triple over $\Theta$ is defined to be a triple $(\Omega, P, \Gamma)$ where $\Omega$ is a finite set, $P$ is a probability function on $\Omega$, and $\Gamma$ is a function from $\Omega$ to $2^\Theta$, such that for all $\omega \in \Omega$, $\Gamma(\omega) \neq \emptyset$ and $P(\omega) \neq 0$.

Associated with a source triple is a mass function, and hence a belief function, given respectively by $m(X) = \sum_{\omega: \Gamma(\omega) \subseteq X} P(\omega)$ and $Bel(X) = \sum_{\omega: \Gamma(\omega) \subseteq X} P(\omega)$. Conversely, any mass/belief function can be expressed in this way for some (non-unique) source triple. Each belief function (or source triple) is intended to represent a separate piece of evidence. The impact of a set of independent evidences is calculated using Dempster's rule, which (in terms of source triples) is intended to represent a separate piece of evidence.

The time that the algorithm takes to achieve a given accuracy is roughly proportional to $|\Theta| m/P'(\Omega)$, making it very efficient for problems where the evidences are not very conflicting (Wilson 1991).

If, however, there is high conflict between the evidences, so that $P'(\Omega)$ is extremely small, then it will tend to take a very long time to find an $\omega$ in $\Omega$.

Example Let $\Theta = \{x_1, x_2, \ldots, x_m\}$, for each $i = 1, \ldots, m$ let $\Omega_i = \{1, 2\}$, let $P_i(1) = P_i(2) = 1/2$, let $\Gamma_i(1) = \{x_i\}$ and let $\Gamma_i(2) = \Theta$. The triple $(\Omega_i, P_i, \Gamma_i)$ corresponds to a simple support function (see (Shafer 1976)) with $m_i(\{x_i\}) = 1/2$ and $m_i(\Theta) = 1/2$. The conflict between the evidences is very high for large $m$ since we have $P'(\Omega) = (m + 1)/2^m$ so the simple Monte-Carlo algorithm is not practical.

### 3. A Simple Monte-Carlo Algorithm
Since, for $X \subseteq \Theta$, $Bel(X) = PDS(\Gamma(\omega) \subseteq X)$, the obvious idea for a Monte-Carlo algorithm for calculating $Bel(X)$ is to repeat a large number of trials, where for each trial, we pick $\omega$ with chance $PDS(\omega)$ and let the value of the trial be 1 if $\Gamma(\omega) \subseteq X$, and 0 otherwise. $Bel(X)$ is then estimated by the average value of the trials. We can pick $\omega$ with chance $PDS(\omega)$ by repeatedly (if necessary) picking $\omega \in \Omega$ with chance $P_i(\omega)$ until we get an $\omega$ in $\Omega$. (Picking $\omega$ with chance $P'(\omega)$ is easy: for each $i = 1, \ldots, m$ we pick $\omega_i \in \Omega_i$ with chance $P_i(\omega_i)$ and let $\omega = (\omega_1, \ldots, \omega_m)$.)

The time that the algorithm takes to achieve a given accuracy is roughly proportional to $|\Theta| m/P'(\Omega)$, making it very efficient for problems where the evidences are not very conflicting (Wilson 1991).

If, however, there is high conflict between the evidences, so that $P'(\Omega)$ is extremely small, then it will tend to take a very long time to find an $\omega$ in $\Omega$.

### 4. The Markov Chain Monte-Carlo Algorithms
Here we 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.

#### 4.1 The Connected Components of $\Omega$

The Markov Chain algorithms that we will consider require a particular condition on $\Omega$ to work, which we will call connectedness. This corresponds to the Markov Chain being irreducible (Feller 1950).

For $i \in \{1, \ldots, m\}$ and $\omega, \omega' \in \Omega$ write $\omega \equiv_i \omega'$ if $\omega$ and $\omega'$ differ at most on their ith co-ordinate, i.e.,

---

1. It also seems to be more convenient for justification of Dempster's rule, see (Shafer 1981, Wilson 1993).
2. There has been much discussion in the literature on the soundness of the rule, e.g., (Pearl 1990, IJAR 1992); justifications include (Shafer 1981, Ruspini 1987, Wilson 1989, 1993).
if for all \( j \in \{1, \ldots, m\} \setminus \{i\}, \omega(j) = \omega'(j) \). Let \( R \) be the union of the relations \( \equiv_i \) for \( i \in \{1, \ldots, m\} \), so that \( \omega R \omega' \) if and only if \( \omega \) and \( \omega' \) differ at most on one co-ordinate; let equivalence relation \( \equiv \) be the transitive closure of \( R \). The equivalence classes of \( \equiv \) will be called connected components of \( \Omega \), and \( \Omega \) will be said to be connected if it has just one connected component, i.e., if \( \equiv \) is the relation \( \Omega \times \Omega \).

### 4.2 The Basic Markov Chain Monte-Carlo Algorithm

Non-deterministic function \( \text{PDS}^N(\omega_0) \) takes as input initial state \( \omega_0 \in \Omega \) and number of trials \( N \) and returns a state \( \omega \). The intention is that when \( N \) is large, for any initial state \( \omega_0 \), \( \Pr(\text{PDS}^N(\omega_0) = \omega) \approx \Pr(\text{DS}(\omega)) \) for all \( \omega \in \Omega \). The algorithm starts in state \( \omega_0 \) and randomly moves between elements of \( \Omega \). The current state is labelled \( \omega_c \).

**FUNCTION PDS**

\[
\omega_c := \omega_0 \\
\text{for } n = 1 \text{ to } N \\
\quad \text{for } i = 1 \text{ to } m \\
\quad \quad \omega_c := \text{operation}_i(\omega_c) \\
\quad \text{next } i \\
\text{next } n \\
\text{return } \omega_c.
\]

Non-deterministic function \( \text{operation}_i \) changes at most the \( i \)th co-ordinate of its input \( \omega_c \)—it changes it to \( y \) with chance proportional to \( P_i(y) \). We therefore have, for \( \omega, \omega' \in \Omega \),

\[
\Pr(\text{operation}_i(\omega') = \omega) = \left\{ \begin{array}{ll}
\alpha_{\omega'} P_i(\omega(i)) & \text{if } \omega \equiv_i \omega' \\
0 & \text{otherwise}.
\end{array} \right.
\]

The normalisation constant \( \alpha_{\omega'} \) is given by \( \alpha^{-1}_{\omega'} = \sum_{\omega \equiv_i \omega'} P_i(\omega(i)) \).

### 4.3 The Calculation of Belief

Now that we have a way of picking \( \omega \) with chance approximately \( \Pr(\text{DS}(\omega)) \) we can incorporate it, in the obvious way, in an algorithm for calculating \( \text{Bel}(X) \). This gives function \( \text{B}^N_K(\omega_0) \) with inputs \( \omega_0, N \) and \( K \), where \( \omega_0 \in \Omega \) is a starting value, \( N \) is the number of trials, and \( K \) is the number of trials used by the function \( \text{PDS}^K(\cdot) \) used in the algorithm. The value \( \text{B}^N_K(\omega_0) \) can be seen to be the proportion of the \( N \) trials in which \( \text{TS}(\omega) \subseteq X \).

In the \( \text{B}^N_K(\omega_0) \) algorithm, for each call of \( \text{PDS}^K(\cdot) \), \( K \) new values of \( \omega \) are generated, but only one, the last, is used to test if \( \text{TS}(\omega) \subseteq X \). It may well be more efficient to use all of the values, which is what \( \text{BEL}^N(\omega_0) \) does. The implementation is very similar to that for \( \text{PDS}^N(\omega_0) \), the main difference being the extra line in

\[
\text{FUNCTION } \text{BEL}^N(\omega_0) \\
\quad \omega_c := \omega_0 \\
\quad S := 0 \\
\quad \text{for } n = 1 \text{ to } N \\
\quad \quad \omega_c := \text{PDS}^N(\omega_c) \\
\quad \quad \text{if } \text{TS}(\omega_c) \subseteq X \\
\quad \quad \quad \quad \quad \quad S := S + 1 \\
\quad \text{next } n \\
\text{return } \frac{S}{N}.
\]

The key result is the following.

**Theorem** Suppose \( \Omega \) is connected. Then

1. given \( \varepsilon > 0 \) there exists \( N' \) such that for all \( N \geq N' \), any \( \omega \in \Omega \) and any starting value \( \omega_0 \), \( |\Pr(\text{PDS}^N(\omega_0) = \omega) - \Pr(\text{DS}(\omega))| < \varepsilon \);
2. given \( \varepsilon, \delta > 0 \) there exists \( K' \) and \( N' \) such that for all \( K > K' \) and \( N > N' \) and any \( \omega_0 \), \( \Pr(|\text{B}^N_K(\omega_0) - \text{Bel}(X)| < \varepsilon) \geq 1 - \delta \); and
3. given \( \varepsilon, \delta > 0 \) there exists \( N' \) such that for all \( N > N' \) and any \( \omega_0 \), \( \Pr(|\text{BEL}^N(\omega_0) - \text{Bel}(X)| < \varepsilon) \geq 1 - \delta \).

This shows that \( \text{PDS}^N \) approximates \( \text{DS} \) to arbitrary accuracy, and that \( \text{B}^N_K \) and \( \text{BEL}^N \) approximate \( \text{Bel}(X) \) to arbitrary accuracy. The proof of this theorem is a consequence of general convergence results for Markov Chain Monte-Carlo algorithms; a summary of these can be found in (Smith & Roberts 1993). However, the main problem with the convergence results is that, in general, it is very difficult to assess when we have reached the desired precision.

The reason that we require that \( \Omega \) be connected is that, in the algorithms, the only values that \( \omega_c \) can take are the members of the \( \equiv \)-equivalence class of the starting position \( \omega_0 \).

### 4.4 Speeding up Intersections

In the implementation of \( \text{B}^N_K \) and \( \text{BEL}^N \) we have to perform \( \text{operation}_i(\omega) \), which involves changing the \( i \)th co-ordinate of \( \omega \) to \( y \), with a probability proportional to \( P_i(y) \). The main difficulty lies in that the new \( \omega' \) has to belong to \( \Omega \), that is, \( \text{TS}(\omega') \neq \emptyset \). In order to find possible values of \( \omega' \), we have to calculate the intersection of all the sets \( \text{TS}(\omega_j) \) for \( j \neq i \). This calculation is of order \( O(|\Theta|(m - 1)) \).

However, this operation can be speeded up. Define, for each \( \omega \in \Omega \), a function \( h_\omega : \Theta \rightarrow \mathbb{N} \) given

\[
h_\omega(\theta) = \min \{ j : \theta(j) \neq \omega(j) \}.
\]
by \( h_\omega(\theta) = \sum_{i=1}^{m} I_{\Gamma_i(\omega_i)}(\theta) \) where \( I_{\Gamma_i(\omega_i)}(\theta) \) is equal to 1 if \( \theta \in \Gamma_i(\omega_i) \) and 0 otherwise. Suppose we have stored the function \( h_\omega \). If we calculate \( h_\omega - I_{\Gamma_i(\omega_i)} \) then the desired intersection is given by the elements of \( \Theta \) with a maximum value of this difference.

Suppose now that we have randomly picked a new \( \omega \)th co-ordinate, \( \psi \). We can calculate the new \( h \)-function, \( h_\psi \), by just adding \( I_{\Gamma_i(\psi_i)} \) to the above difference. This method allows us to calculate the intersection in \( O(|\Theta|) \).

5. The Connectivity of \( \Omega \)

For the above algorithms \( B^N_k \) and \( \text{BEL}^N \) to converge we require that \( \Omega \) be connected. Many important cases lead to a connected \( \Omega \); for example, if the individual belief functions \( \text{BEL}_i \) are simple support functions, consonant support functions, discounted Bayesian or any other discounted belief functions (i.e., with \( m_i(\Theta) \neq 0 \)) then \( \Omega \) will be connected. Other cases clearly lead to non-connected \( \Omega \), for example, if each \( \text{BEL}_i \) is a Bayesian belief function. Unfortunately it will sometimes not be at all clear whether \( \Omega \) is connected or not, and the obvious way of testing this requires a number of steps exponential in \( m \). In 5.1 we construct a method for dealing with this problem.

5.1 Using \( \Theta \) to Find the Connected Components

\( \Theta \), the core of \( \Theta \), is defined to be \( \bigcup_{\omega \in \Omega} \Gamma(\omega) \). For \( \theta \in \Theta \), let \( \theta^* \subseteq \Omega \) be the set \( \{\omega \in \Omega : \Gamma(\omega) \ni \theta \} \) and, for \( i \in \{1, \ldots, m\} \), \( \theta_i^* = \{\omega_i \in \Omega_i : \Gamma_i(\omega_i) \ni \theta \} \).

Define relation \( R' \) on \( \Theta \) by \( \theta R' \psi \iff \omega R \omega' \) for some \( \omega \in \theta^* \) and \( \omega' \in \psi^* \) (relation \( R \) was defined at the beginning of section 4). Let equivalence relation \( \equiv' \) be the transitive closure of \( R' \).

Proposition

(i) Suppose \( \theta, \psi \in \Theta \). Then \( \theta R' \psi \iff \exists \) at most one \( i \) such that \( \theta_i^* \cap \psi_i^* \) is empty.

(ii) A one-to-one correspondence between the equivalence classes of \( \equiv' \) and \( \equiv \) is given by \( X \to \bigcup_{\theta \in X} \theta^* \), for \( \equiv' \)-equivalence class \( X \).

The inverse of this mapping is given by \( W \to \bigcup_{\omega \in W} \Gamma(\omega) \), for \( \equiv \)-equivalence class \( W \).

Part (i) implies that \( R' \) can be expressed easily in terms of commonality\(^4\) functions: \( \theta R' \psi \iff Q_i(\{\theta, \psi\}) = 0 \) for at most one \( i \in \{1, \ldots, m\} \).

The most important consequence of this proposition is that it gives an alternative method for testing if \( \Omega \) is connected: we can use (i) to construct the equivalence classes of \( \equiv' \); by (ii), \( \Omega \) is connected if and only if there is a single \( \equiv' \)-equivalence class. Often \( \Theta \) will be much smaller than \( \Omega \), so this method will be much more efficient than a straightforward approach.

5.2 Finding a Starting Position \( \omega_0 \in \Omega \)

The algorithms require as input a value \( \omega_0 \in \Omega \) (and any element will do). It might seem hard to find such an element \( \omega_0 \) if \( \Omega \) is very much smaller than \( \Omega \). However, we should have no problem in picking an element \( \theta \) in \( \Theta \), the core of \( \Theta \), since the core consists of possibilities not completely ruled out by the evidence. But then, for \( i = 1, \ldots, m \), we can pick \( \omega_i \) such that \( \Gamma_i(\omega_i) \ni \theta \). Letting \( \omega_0 = (\omega_1, \ldots, \omega_m) \), we have \( \Gamma(\omega_0) \ni \theta \) so \( \omega_0 \in \Omega \) as required.

5.3 Barely Connected \( \Omega \)

The convergence theorem guarantees that the algorithms will converge to the correct value for connected \( \Omega \), but it does not say how quickly. The following example illustrates that the convergence rate will tend to be very slow if \( \Omega \) is only barely connected (i.e., if it is very hard for the algorithm to move between some elements of \( \Omega \)).

Example

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

Suppose we want to use function \( B^N_k(\omega_0) \) or function \( \text{BEL}^N(\omega_0) \) to estimate \( \text{BEL}(\{x_1\}) \) (which is just under \( \frac{1}{2} \)). If we start with \( \omega_0 \) such that \( \Gamma(\omega_0) = \{x_1\} \) then it will probably take of the order of \( 2^k \) values of \( \omega \) to reach a member of \( \{x_2\}^* \). Therefore if \( k \) is large, e.g., \( k = 30 \), and we do a million trials then our estimate of \( \text{BEL}(\{x_1\}) \) will almost certainly be 1. Other starting positions \( \omega_0 \) fare no better.

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

6. Experimental Testing

The performance of the three Monte-Carlo algorithms for estimating \( \text{BEL}(X) \), \( \text{MCBEL}^N \), \( B^N_k \) and \( \text{BEL}^N \), was
tested experimentally; MCBEL^N is the simple Monte-Carlo algorithm described in section 3, where \( N \) is the number of times an element \( \omega \in \Omega \) is picked with chance \( P'(\omega) \) (so the number of useful trials will be approximately \( NP'(\Omega) \)).

We considered randomly generated belief functions on a frame \( \Theta \) with 30 elements. The number of focal elements (i.e., sets with non-zero mass) was chosen using a Poisson distribution with mean 8.0. Each focal element \( A \) was determined by first picking a random number \( p \) in the interval \([0, 1]\), and then, independently for each \( \theta \in \Theta \), including \( \theta \) in \( A \) with chance \( p \).

Two experiments were carried out. In the first one, six belief functions were combined and a set \( X \subseteq \Theta \) was randomly generated. The exact belief of this event was calculated and 100 approximations were made for each of the three Monte-Carlo algorithms, with \( N = 5000 \), \( K = 6 \). This was repeated for ten different combinations and ten randomly selected sets \( X \). The second experiment was very similar: the only difference being that 10 belief functions were combined and \( K = 10 \).

The calculation times of the different algorithms were similar. In the light of the results we can conclude:

- The Markov Chain Monte-Carlo algorithms performed significantly better than the simple Monte-Carlo algorithm; e.g., in the first experiment, the mean errors of the Markov Chain algorithms were typically about 0.005, whereas in the simple algorithm, mean errors were typically about 0.01.

- There was no significant difference between the performance of the two Markov Chain Monte-Carlo algorithms. In BEL^N we use more cases of the sample, but in \( B^K_N \) there is a greater degree of independence between the cases.

It also appears that when \( m \) is increased, the relative precision of the Markov Chain algorithms with respect to the simple algorithm increases. This is due to the fact that the degree of conflict increases with \( m \).

Detailed results will appear in (Moral & Wilson 1994).

7. Extensions and Applications

Calculation of Belief on Logics: Dempster-Shafer theory can easily be extended to logics (see also (Pearl 1990, Wilson 1991)). For example, let \( L \) be the language (i.e., the set of well-formed formulae) of a propositional calculus. To extend the definitions of mass function, belief function, source triple and Dempster’s rule we can (literally) just replace the words ‘over \( \Theta \)’ by ‘on \( L \)’, replace \( \mathbb{2}^\Theta \) by \( L \), replace \( \emptyset \) by \( \bot \), \( \subseteq \) by \( \models \) and intersection \( \cap \) by conjunction \( \land \). To adapt the algorithms we just need to change the condition \( \Gamma(\omega_e) \subseteq X \) to the condition \( \Gamma(\omega_e) \models X \) in the functions \( B^K_N(\omega_0) \) and \( BEL^N(\omega_0) \).

**Infinite frames**: The Monte-Carlo algorithms open up the possibility of the computation of Dempster-Shafer belief on infinite frames \( \Theta \), with perhaps also an infinite number of focal elements (so that \( \Omega \) is infinite). Clearly we will need some effective way of intersecting the focal elements; for example, this may be practical if \( \Theta \subseteq \mathbb{R}^n \) for some \( n \), and the focal elements are polytopes.

The algorithms can also be used for calculating Dempster-Shafer belief in belief networks, and in decision-making, for calculating upper and lower expected utility, see (Moral & Wilson 1994).

8. Discussion

Although the Markov Chain Monte-Carlo algorithms appear to often work well, there remains the problem of cases where \( \Omega \) is not (or is barely) connected. We will briefly discuss ways in which the algorithm could be improved for such cases.

**Blocking Components**: A technique sometimes useful in Gibbs samplers is to block together components (see (Smith & Roberts 1993)); for our problem, this amounts to changing simultaneously more than one co-ordinate of \( \omega \) at a time; this can connect up components which were previously not connected (but will increase the time for each trial).

**Artificially Increasing \( \Omega \)**: Recall that the state space \( \Omega \) was defined to be \( \{ \omega \in \Omega : \Gamma'(\omega) \neq \emptyset \} \). If we use \( \Omega' \) as the state space where \( \Omega \subseteq \Omega' \subseteq \Omega \), and define relations \( \equiv_\Omega \), \( \equiv \) and \( \equiv_\Omega' \), then the functions \( B^K_N(\omega_0) \) and \( BEL^N(\omega_0) \) will converge to \( Bel(X) \), given that \( \Omega' \) is connected, so long as we don’t count trials where \( \omega \in \Omega' \setminus \Omega \) (i.e., we don’t increment \( S \) or the trial counter for such an \( \omega \)). This means that if \( \Omega \) is not connected or is barely connected then we could improve the connectivity by judiciously adding extra points to \( \Omega \).

**Weighted Simulation**: Another approach to solving this type of problem, is to sample from a wrong but easier model and weighting to the distribution of interest (also known as ‘importance sampling’). This method has been used in Bayesian networks (Fung &

---

\(^5\)Note that now a belief function does not determine a unique mass function.

These definitions also work for many other logics, such as modal logics, or \( L \) could be the set of closed formulae in a first order predicate calculus.
Chang 1990). It could be used directly, or in conjunction with the algorithms given here to improve the connectivity of Ω.

Several combinations of the above procedures could produce optimal results in difficult situations. A simple strategy is to combine first by an exact method the groups of belief functions with a high degree of conflict, and then to combine the results with a simulation procedure.

Acknowledgements

The second author is supported by a SERC postdoctoral fellowship. This work was also partially supported by ESPRIT (I and II) basic research action 3085, DRUMS. We are also grateful for the use of the computing facilities of the school of CMS, Oxford Brookes University.

References


IJAR, 92, *International Journal of Approximate Reasoning*, 6, No. 3 [special issue].


