BeliefNet Tool: An Evidential Network Toolbox for Matlab

Walid Trabelsi
LARODEC-ISG
ISG Tunis, Tunisia
wsacraf@gmail.com

Boutheina Ben Yaghlane
LARODEC-ISG
IHEC Carthage, Tunisia
boutheina.yaghlane@ihec.rnu.tn

Abstract

The *BeliefNet Tool* is a new engine to perform local computations efficiently and conveniently in both undirected and directed evidential networks (i.e. networks using belief functions theory [10], [12]). *BeliefNet Tool* is written in Matlab and it implements the basic algorithms as well as the main operations performing inference of beliefs in evidential networks. Attention is especially devoted to present the structures used when implementing *BeliefNet Tool*.

Keywords: Belief functions, Conditional belief functions, Directed evidential network, Binary join tree.

1 Introduction

We can appreciate the great number of applications dealing with the field of quantitative reasoning and decision under uncertainty with the help of Bayesian networks [9]. The emergence of such applications is expected since inconsistent, inaccurate and uncertain information represent a real-life aspect of human beings. The massive use of Bayesian networks is justified by their acceptable computational complexity. This deployment of Bayesian networks can not hide their inadequacy for some situations involving ignorance. In contrary, representing ignorance is possible with the Dempster-Shafer (DS)'s belief functions (BF) theory but with higher computational complexity. So, tools dealing with DS's theory will be of great help for researchers in this field.

This paper describes one attempt to build such tool, called the *BeliefNet tool*. So the main goal of this paper is to present the design and the implementation of *BeliefNet Tool* dealing with inference in both undirected and directed evidential networks (EN).

The remain of this paper is organized as follows. Section 2 presents the graphical representation of EN. Section 3 describes briefly the algorithms performing exact inference in both singly-connected and multi-connected EN. Section 4 details the features and the design of *BeliefNet tool* by presenting the structures used in this tool. Section 5 details the basic operations for inference in EN. Section 6 presents practical issues of *BeliefNet Tool*. Finally, we conclude with some future perspectives.

2 Evidential Networks Representation

A graphical model is considered as a picture that provides an intuitive description of the problem. It is considered also as a mathematical structure that specifies the different connections between the variables of a problem transforming a complex problem into an easy and clear representation. An evidential network (EN) is a graphical model quantified by the means of belief functions (BF). There are two kinds of EN: *undirected EN* and *directed EN*. The parametrization of each one will be discussed in the following sections.

2.1 Parametrization of Directed Models

A directed model can be parameterized by specifying the elements of the problem regarded as random variables. These variables are represented explicitly as nodes. The relations between the specified variables are determined in a mapping set between variables. Once the qualitative part of the EN is specified, the quantitative part is specified by the means of *prior* and *conditional* BF distributions, i.e, the distribution $bel(X_i|Pa_i)$, where X_i represents node i and Pa_i are its parents. These distributions will be endowed to concerned nodes. The variables can be represented as integers. The distributions can be represented as tables. These tables are simple to manipulate but have the disadvantage of requiring a number of parameters that is exponential with the number of parents.

2.2 Parametrization of Undirected Models

Undirected graphical models, also known as Markov networks, are common in the physics and computer vision communities. The elements of the problem are sets of variables. Every set of variables is represented as an ordered sequence of variables in a clique. The distributions are quantified through *joint* BF and they are named clique potentials. If all variables are discrete these potentials can be represented as tables. If a clique contains n binary variables it requires 2^n parameters (a large number of parameters).

3 Exact Inference in Evidential Networks

In [13], the author introduces the notion of directed EN with conditional belief functions (CBF) in which two rules are proposed, namely the Generalized Bayesian Theorem (GBT) and the Disjunctive Rule of Combination (DRC). In [1], the author proposed two algorithms for propagation in *singly-connected* EN[1] based on these rules:

- The first one is an extension of Mellouli's algorithm (already proposed for undirected EN [7]) for directed EN. The main idea of this algorithm is to represent concerned variables as nodes connected by links (or edges) according to (in)dependence relations in the initial structure. The edges represent "railways" when nodes communicate with each other by messages. These messages are conceived by applying the GBT and DRC [13] depending on whether the sender is the child or the parent of the receiver. This algorithm is based on the technique of *local computations* [5].
- The second one is based on Pearl's algorithm as described in [9] and [8]. It can be described as two steps: *belief initialization* and *belief updating* doing efficient computations by a message-passing scheme using the GBT and the DRC rules.

For multi-connected EN, two algorithms can be found in the literature:

- The first one uses a *binary join tree* (BJT) for representing and inferring beliefs in networks (using joint belief functions) [11]. The BJT is constructed by a process having as a main idea the *fusion algorithm*: a successive variable elimination. The belief propagation scheme through the BJT involves two phase-propagation, namely a *propagation-up* and a *propagation-down.*
- The second one proposes a *modified binary join tree* (MBJT) (maintaining conditional belief functions) [2]. The basic idea is to transform the initial directed EN into a BJT in which some modifications are proposed. The obtained graphical structure, called a MBJT, emphasizes

[1]Also called polytrees, they are graphs where no more than one (undirected) path connects every two nodes.

explicitly the conditional relations. In order to avoid the computations of joint belief functions as required when using the fusion algorithm in BJT, the DRC and the GBT are used for making inference efficiently in the MBJT.

All these algorithms have been implemented in our *BeliefNet Tool*.

4 Implementation Aspects

In the following, we will present the implementation choices in order to perform efficient computations of marginals in directed and undirected EN. All operations executed during inference process (combination, marginalization, etc) deal with mass functions. When we say mass functions, we invoke focal sets of a mass function and their masses. Inappropriate representation of focal sets provokes extra computations and needs much more memory space in a computer. When BF theory is concerned, the loss of time and memory space are much more heavy than when probability theory is concerned. Indeed, contrary to probability theory which need to store and assign a mass to one state of a system (i.e. singleton), the BF theory stores and assigns one mass to more than one state of the system.

4.1 Representing Mass Functions

Mass functions are generally described as follows:

$$m : 2^{\Theta} \longrightarrow [0,1]$$
$$A \longrightarrow v$$

A mass function m associates a mass value v to a focal element A. Θ is the frame of discernment and A is a subset of Θ.

[6] showed that a mass function is completely determined by its focal sets and their relative masses.

4.2 Representing Focal Sets

[3] showed that focal elements represent the core supporting belief masses. Generally, they are regarded as sets of variable configurations. Basic operations on focal sets are projection and extension. These operations manipulate intensively focal sets during the inference process. This justifies the influence of focal sets encoding on the performance of the inference process.

After making the comparison between different representations (bitmasks, binary representation, disjunctive normal forms, etc), [6] argues that the binary representation of focal sets is the best choice to perform efficient computations with BF.

Therefore, our implementation of the propagation framework will adopt the binary representation.

4.2.1 Binary Representation

A focal element, as a set of configurations, is represented by a *bitset*. Each bit of the bitset corresponds to a *configuration*: if the bit is set to 1 then the configuration is present in the focal element else (i.e. the bit is set to 0) the configuration is not present in the focal element. This supposes a global ordering of the involved variables.

Suppose that we have a mass function S on domain $D = \{x_1, ..., x_n\}$. Let $\Theta_{x_i} = \{x_{i1}, ..., x_{in}\}$ be the set of values of the variable x_i. A configuration on Θ_D is a vector $(v_1, ..., v_n)$ such that v_i is a value from Θ_{x_i}. A focal set F of a mass function S is a subset of Θ_D; in other words, F represents a set of configurations of values of Θ_D. Let $C = \{c_1, ..., c_m\}$ be the set of all possible configurations on D.

$|C| = \prod_{i=1:n}(\Theta_{x_i}), D = \{x_1, ..., x_n\}$ and x_i has its values in Θ_{x_i}. Therefore, F can be represented by a bit vector V_F such that:

V_F=$[b_1\ b_2 ..., b_m]$ *where* $\{b_i = 1\ if\ c_i \in F,\ otherwise\ \ b_i = 0\}$

Example 1 Binary representation of focal sets.

Suppose $D = \{x_1, x_2, x_3\}$.

Let $\Theta_{x_1} = \{x_{11}, x_{12}\}$, $\Theta_{x_2} = \{x_{21}, x_{22}, x_{23}\}$ and $\Theta_{x_3} = \{x_{31}, x_{32}, x_{33}\}$ be the sets of values of x_1 , x_2 and x_3, respectively.

$F_1 = \{(x_{11}, x_{22}, x_{31}), (x_{11}, x_{22}, x_{33})\}$

$F_2 = \{(x_{11}, x_{21}, x_{32})\}$

$C = \{(x_{11}, x_{21}, x_{31}); (x_{11}, x_{21}, x_{32}); (x_{11}, x_{21}, x_{33})$
$; (x_{11}, x_{22}, x_{31}); (x_{11}, x_{22}, x_{32}); (x_{11}, x_{22}, x_{33})$
$; (x_{11}, x_{23}, x_{31}); (x_{11}, x_{23}, x_{32}); (x_{11}, x_{23}, x_{33});$
$(x_{12}, x_{21}, x_{31}); (x_{12}, x_{21}, x_{32}); (x_{12}, x_{21}, x_{33});$
$(x_{12}, x_{22}, x_{31}); (x_{12}, x_{22}, x_{32}); (x_{12}, x_{22}, x_{33});$
$(x_{12}, x_{23}, x_{31}); (x_{12}, x_{23}, x_{32}); (x_{12}, x_{23}, x_{33}); \}$

$|C| = 2 * 3 * 3 = 18 \quad configurations$

According to binary representation, F_1 and F_2 are represented by the two following bit vectors:

$V_{F_1} = [0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

$V_{F_2} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$.

Notice that we store only focal sets with positive masses (pairs (F_i, m_i) where F_i is a focal element of Θ_D and m_i is the corresponding positive belief mass $(m_i > 0)$) because if we store all possible focal sets of Θ_D, the binary representation will not be efficient as we store a great number of records which will not be involved in computations. For example if we have $m = 18$ *configurations* as seen in the previous example, then we have to store $2^{18} = 262144$ pairs of (F_i, m_i).

4.3 Representing Potentials

The most high level structures manipulated in DS framework are *potentials.* They store focal elements and their corresponding belief masses (F_i, m_i). During the inference process, combination and marginalization make heavy use of potentials which explains the influence of the potential representation on computations.

In [6], the author exposed different structures. Potentials can be represented by *lists.* Manipulating records in this list costs $O(n^2)$ elementary operations where n is the size of the bitset. Another structure was described in [3], called the *hash table*, which is a data structure used to store pairs (key, value). It provides a *hash function* to navigate, access, modify efficiently a value by use of the key. The advantage of this data structure is that the key can be an element of the information stored which makes easy and intuitive the manipulation of stored information. Manipulating records in this table costs $O(n^2/s)$ elementary operations where n is the size of the bitset and s is the size of the table (the number of records).

The operations of coarsening and aggregation of knowledge during inference process make heavy use of computations and memory space when manipulating mass functions. After performing tests, we chosen the structure of *hash table* as structure to store the mass functions (or potentials).

4.4 Representing Join Tree

When BF theory is concerned, the join tree is regarded as a number of nodes connected by the means of joint belief functions (for the BJT) or conditional belief functions (for the MBJT). Every node is a cluster of variables. Variables are represented by *integers.* Sets of variables are represented by *ordered variables.* Thus, the mass function will have focal sets defined on ordered domains. The join tree (BJT or MBJT) is represented as a mapping set between ordered sequences of variables. Every edge in the join tree is represented by an element among the mapping set. This will improve the efficiency of the basic operations.

5 Basic Operations for Belief Propagation

The basic operations for belief functions theory are combination and marginalization. These two operations invoke two other operations: extension and projection. We should notice that the complexity associated to these operations depends on the representation of potentials. The complexity of operations will be compatible with the *hash table.*

Combination: $(bel_1, bel_2) \rightarrow bel_1 \oplus bel_2$.

If bel_1 and bel_2 are potentials on D_1 and D_2 respectively, $bel_1 \oplus bel_2$ is a potential on $D = D_1 \cup D_2$.

domain$(bel_1 \oplus bel_2) = domain(D_1) \cup domain(D_2)$. Let $D = D_1 \cup D_2$, F_1 and F_2 be two focal sets defined on D_1 and D_2 respectively and m_1 and m_2 are the corresponding masses. The mass m of a focal set F equivalent to the intersection of the extensions of F_1 and F_2 is obtained such that:

$$m = \sum_{F_1^{\uparrow D} \cap F_2^{\uparrow D} = F} \{m_1 * m_2\} \tag{1}$$

Marginalization: $(bel, D') \rightarrow bel^{\downarrow D}$.

Let F be a focal set defined on Θ_D. If bel is a potential for D and $D' \subseteq D$, then bel' is a potential for D'. Therefore domain $(bel') = D'$. Let m be the mass of F and m' be the mass of the focal set F' in the new potential bel'.

$$m' = \sum_{F^{\downarrow D'} = F'} \{m\} \tag{2}$$

When we observe the formulas (1) and (2), we understand that the operations of extension and projection are the key operations. So, the mass functions representation is determined in order to make these two operations more efficient. In the following, we will draw up the corresponding algorithms.

Projection: To project a focal set F is to change the domain d on which this focal set is defined. The new domain D will determine the new configurations ($newconfig$) of the new focal set (F_{res}). The algorithm navigates through the configurations g of F and eliminates the values of eliminated variables found in $\{D - d\}$. Thus, every configuration g will see a number of its elements deleted using the function *Truncate*. The function *Projection* returns the new configurations in (F_{res}) (see example 2).

Complexity of projection: $O(n)$ where n is the number of variables to eliminate.

```
Projection(F , D)
$F_{res} \leftarrow \emptyset$
$d \leftarrow Domain(F)$
If $d = D$
    $F_{res} \leftarrow F$
else
    $focus \leftarrow d \cap D$
    For each $configuration \quad g \in F$
        $newconfig \leftarrow Truncate(g, focus)$
        $F_{res} \cup (newconfig)$
    End For
End If
$return(F_{res})$
```

Extension: During the extension process, the algorithm adopts the same approach as in the *Projection* function, but when extending a focal set F we will add a number of elements to every configuration g. Therefore, g may be extended into more than one configuration using the function *Extend* according to the values of variables found in *extent* $(d \cap D)$ (see example 2).

```
Extension(F , D)
$F_{res} \leftarrow \emptyset$
$d \leftarrow Domain(F)$
If $d = D$
    $F_{res} \leftarrow F$
else
    $extent \leftarrow d \cap D$
    For each $configuration \quad g \in F$
        $newconfig \leftarrow Extend(g, extent)$
        $F_{res} \cup (newconfig)$
    End For
End If
$return(F_{res})$
```

Complexity of extension: $O(n)$ where n is the number of variables to add.

We note that for the two operations projection and extension, we verify whether the domain d of the focal set F to project or to extend is equal or not to the destination domain D. This will avoid extra effort since the focal set F will remain the same.

We notice that the function Extend and Truncate are built-in function in Matlab.

Example 2 Extension and Projection of focal sets.

Let $D = \{x_1, x_2, x_3\}$. Let $\Theta_{x_1} = \{x_{11}, x_{12}\}$, $\Theta_{x_2} = \{x_{21}, x_{22}, x_{23}\}$ and $\Theta_{x_3} = \{x_{31}, x_{32}, x_{33}\}$ be the sets of values of x_1 , x_2 and x_3 respectively. Let F_1 and F_2 be two focal sets defined respectively on $D_1 = \{x_1, x_2\}$ and $D_2 = D$.

$F_1 = \{(x_{11}, x_{22})\}$

$F_2 = \{(x_{11}, x_{21}, x_{31}); (x_{12}, x_{21}, x_{31}); (x_{12}, x_{22}, x_{33})\}$

Suppose we will extend F_1 to be defined on D, and project F_2 to be defined on $D_3 = \{x_2, x_3\}$.

Let $c_1 = (x_{11}, x_{22})$ be a configuration of F_1. So, extent=x_3 defined on $\Theta_{x_3} = \{x_{31}, x_{32}, x_{33}\}$ So, $c_1' \leftarrow Extend(c_1, extent)$ thus $F_1^{\uparrow D} = \{(x_{11}, x_{22}, x_{31}); (x_{11}, x_{22}, x_{32}); (x_{11}, x_{22}, x_{33})\}$. We deduce that the function Extend expanded c_1 into 3 configurations corresponding to the 3 values of x_3.

$F_2^{\downarrow D_3} = \{(x_{21}, x_{31}); (x_{22}, x_{33})\}$. The function Truncate eliminates the values relative to focus=x_1 from configurations in F_2.

We notice that the unique configuration of the focal set F_1 is extended into three new configurations because the variable added to the old domain (x_3) has three values. The number of configurations in the focal set F_2 is reduced from three to two since we eliminated the instances (or values) of the eliminated variables (x_1) from the old configurations. After elimination, we found duplicate configurations which we eliminate in the new focal set $F_2^{\downarrow D_3}$. Once we have presented the core of marginalization and combination, we are able to set the corresponding algorithms.

Let bel_1 and bel_2 be two potentials defined on D_1 and D_2 respectively, and $D = D_1 \cup D_2$. For every potential (bel_1 and bel_2), we navigate through the pairs (F, m) to extend the focal set F into $Fres$ and puts the new pair $(Fres, m)$. Therefore we obtain two potentials in which the focal sets are defined on the same domain D. Then, we will put the intersection between every focal set $Fres_1$ in the potential $Vres_1$ and $Fres_2$ in the potential $Vres_2$ into a new focal set ($newfocalset$) and assign to it the product of the two corresponding masses m_1 and m_2. The pair $(newfocalset, m_1 * m_2)$ is added to a potential $Vinter1$. Since many couples $Fres_1$ and $Fres_2$ may have the same intersections, there will be similar focal sets in pairs $(newfocalset, m_1 * m_2)$ into $Vinter1$. This implies the use of a regrouping function called $Regroup$ (see below). Therefore we obtain a new potential $Vinter2$. This potential will not be the result of the combination of the two initial potentials bel_1 and bel_2 if there was an empty intersection in $newfocalset$ because we should call the $Normalize$ function which will normalize all pairs into $Vinter2$ (see below). As a result we obtain bel_{res}.

```
**Combine**($bel_1$, $bel_2$)
$Vres_1 \leftarrow \emptyset$
If $D_1 = D$
    $Vres_1 \leftarrow bel_1$
else
    For each $pair(F_1, m_1) \in bel_1$
        $Fres_1 \leftarrow Extension(F_1, D)$
        $Vres_1 \cup (Fres_1, m_1)$
    End For
End If
$Vres_2 \leftarrow \emptyset$
If $D_2 = D$
    $Vres_2 \leftarrow bel_2$
    For each $pair(F_2, m_2) \in bel_2$
        $Fres_2 \leftarrow Extension(F_2, D)$
        $Vres_2 \cup (Fres_2, m_2)$
    End For
End If
$Vinter \leftarrow \emptyset$
For each $pair(Fres_1, m_11) \in Vres_1$
    For each $pair(Fres_2, m_22) \in Vres_2$
        $newfocalset \leftarrow Fres_1 \cap Fres_2$
        If $newfocalset \neq \emptyset$
          $Vinter1 \cup (newfocalset, (m_1 * m_2))$
        else
          $emptym \leftarrow emptym + (m_1 * m_2)$
        End If
    End For
End For
$bel_{res} \leftarrow \emptyset$
If $emptym > 0$
    $Vinter2 \leftarrow Regroup(Vinter1)$
    $bel_{res} \leftarrow Normalize(Vinter2, emptym)$
else
    $bel_{res} \leftarrow Regroup(Vinter1)$
End If
$return(bel_{res})$
```

Complexity of combination : $O((n * m)^2/(s * t))$

elementary operations where n and m are respectively the sizes of the bitset stored in potentials bel_1 and bel_2 , and s and t are the sizes of bel_1 and bel_2 (the number of records).

The *Regroup* function will regroup the masses m_2 of focal sets F_2 which have the same configurations. Indeed it will navigate through the pairs (F_1, m_1) of the potential *bel* and sums the masses of focal sets having the same configurations into *summass*. After that *summass* is assigned to a one copy of the similar focal sets to construct a pair $(F_1, summass)$ which will be added to the result potential bel_{res}.

```
Regroup($bel$)
$bel_{res} \leftarrow \emptyset$
For each $pair(F_1, m_1) \in bel$
    $summass \leftarrow 0$
    For each $pair(F_2, m_2) \in bel$
        If $F_1 = F_2$
            $summass \leftarrow summass + m_2$
        End If
    End For
        $bel_{res} \cup (F_1, summass)$
End For
$return(bel_{res})$
```

Complexity of regrouping : $O(n^2/s)$ elementary operations where n is the size of the bitset and s is the size of the potential *bel* (the number of records).

The function $Normalize(P)$ normalizes the potential P according to the closed world assumption [12]. It redistributes the conflict mass associated to the empty focal element after empty intersections during combination.

```
Normalize(bel,emptym)
$bel_{res} \leftarrow \emptyset$
For each $pair(F, m) \in bel$
    $m' \leftarrow m/(1 - emptym)$
    $bel_{res} \cup (F, m')$
End For
$return(bel_{res})$
```

Complexity of normalization : $O(n^2/s)$ elementary operations where n is the size of the bitset and s is the size of the potential *bel* (the number of records).

The marginalization of a potential *bel* on a domain D concerns only the focal sets of *bel*. Indeed, the function $Marginalize$ will navigate through the focal sets F in pairs (F, m) of *bel* and will project F on the domain D to obtain F'. The new focal set F' will have as a mass the same as the one of F and the new pair (F', m) is added to the result potential bel_{res}.

```
Marginalize($bel$ , $D$)
$bel_{res} \leftarrow \emptyset$
For each $pair(F, m) \in bel$
    $F' \leftarrow Projection(F, D)$
    $bel_{res} \cup (F', m)$
End For
$return(bel_{res})$
```

Complexity of marginalization : $O(n^2/s)$ elementary operations where n is the size of the bitset and s is the size of the potential *bel* (the number of records).

6 Software Issues

Aside from the various theoretical issues we have discussed above, there are some practical issues which make the difference between software packages.

6.1 Relevant attributes of BeliefNet Tool

Target people: All available packages are free for academic use.

Availability: The first version of *BeliefNet Tool* will be released soon and will be distributed under the GNU/GPL licence.

Used language: Matlab (see section 6.2)

Extensibility: Since *BeliefNet Tool* aims at academic use it is open to new algorithms and new representations. For instance, we can add approximate algorithms and try to experiment new structures to hold the distributions.

API (application program interface) availability: The new version of Matlab permits to integrate our packages into other applications. So, *BeliefNet Tool* can be used as an autonomous program or as a module in an other application.

6.2 Why Matlab ?

Matlab is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. It provides a good debugger and profiler which make implementation more efficient. Its basic data is an array that doesn't require dimensioning. This allows us to solve many technical computing problems, especially those with matrix and vector formulations.

Matlab is a high-level language which liberates the mind from details like memory management, and enables one to write clear and concise code. Nowadays, Matlab is the most used language for a high productivity research, development, and analysis. One of the incontestable advantages of this language is its family of additional application-specific solutions called *toolboxes*.

7 Conclusion

This paper has introduced, *BeliefNet Tool*, a new DS engine. It described the most significant algorithms of belief propagation in EN. In addition, this paper provided suitable structures and implementation aspects adopted when conceiving *BeliefNet Tool*.

To make *BeliefNet Tool* more useful we plan to implement approximate inference algorithms [4]. In the near future we will integrate the inference algorithms of [14]. We are also interested in making the graphical user interface more interactive and easy to use such as create graphs interactively.

References

[1] Ben Yaghlane, B. (2002). Uncertainty representation and reasoning in directed evidential networks. PhD thesis, Institut Supérieur de Gestion de Tunis Tunisia.

[2] Ben Yaghlane, B. and Mellouli, K. (2006). Belief functions propagation in directed evidential networks. In Proceedings of IPMU'2006, Information Processing and Management of Uncertainty in Knowledge-based Systems, pages 1451–1458, Paris, France.

[3] Haenni, R. and Lehmann, N. (2001). Implementing belief function computations. Technical report, Department of Informatics, University of Fribourg.

[4] Haenni, R. and Lehmann, N. (2002). Resource bounded and anytime approximation of belief function computations. International Journal of Approximate Reasoning, 31:103–154.

[5] Lauritzen, S. and Shenoy, P. (1996). Computing marginals using local computation. Technical Report 267, Kansas University, School of Business.

[6] Lehmann, N. (2001). Argumentation systems and belief functions. PhD thesis, Institut fur Informatik, Fribourg.

[7] Mellouli, K. (1987). On the propagation of beliefs in networks using the Dempster-Shafer theory of evidence. PhD thesis, School of Business, University of Kansas, Lawrence, KS.

[8] Neapolitan, R. E. (1990). Probabilistic reasoning in expert systems: Theory and algorithms. Willey Interscience, New York, NY.

[9] Pearl, J. (1988). Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Pub. San Mateo, Ca, USA.

[10] Shafer, G. (1976). A mathematical theory of evidence. Princeton Univ. Press. Princeton, NJ.

[11] Shenoy, P. P. (1997). Binary join tree. International Journal of Approximate Reasoning, 1(2):250–256.

[12] Smets, P. (1978). Un modèle mathématico-statistique simulant le processus du diagnostic médical. PhD thesis, Université Libre de Bruxelles, (available through University Microlm International, 30–32 Mortimer street, London W1N 7RA, thesis 80–70,003).

[13] Smets, P. (1993). Belief functions: the disjunctive rule of combination and the generalized Bayesian theorem. International Journal of Approximate Reasoning, 9:1–35.

[14] Xu, H. and Smets, P. (1994). Evidential reasoning with conditional belief functions. In Heckerman, D., Poole, D., and Lopez De Mantaras, R., editors, Uncertainty in Artificial Intelligence 94, pages 598–606. Morgan Kaufmann, San Mateo, Ca.