19th International Conference on Information Fusion
Heidelberg, Germany - July 5-8, 2016

A Framework for Efficient Computation of Belief Theoretic Operations

Lalintha G. Polpitiya*, Kamal Premaratne*, Manohar N. Murthi*, Dilip Sarkar†
*Department of Electrical and Computer Engineering
†Department of Computer Science
University of Miami
Coral Gables, Florida, USA
lalintha@umiami.edu, {kamal, mmurthi, sarkar}@miami.edu

***Abstract*—The Dempster-Shafer (DS) theory is a powerful general framework for reasoning under uncertainty. While the strength of the DS theoretic (DST) framework in its ability to handle a wider variety of data imperfections is not in dispute, a major criticism cast towards DST evidential reasoning is the heavy computational burden it entails. If the advantages offered by DS theory is to be fully realized, it is essential that one explores efficient data structures and algorithms that can be used for DST operations and computations. In this paper, we wish to present a novel generalized computational framework for exactly this purpose. We develop three representations — *DS-Vector, DS-Matrix,* and *DS-Tree* — which allow DST computation to be performed in significantly less time. These three representations can also be utilized as tools for visualizing DST models. A new strategy, which we refer to as *REGAP,* which allows *REcursive Generation of and Access to Propositions* is introduced and harnessed in the development of this framework and computational algorithms. The paper also provides a discussion and experimental validation of the utility, efficiency, and implementation of the proposed data structures and algorithms.**

***Index Terms*—Dempster-Shafer theory, belief functions, evidential reasoning, computational frameworks, data structures, algorithms.**

I. Introduction

The Dempster-Shafer (DS) theory of belief functions, also referred to as evidence theory, was first introduced by Dempster [1], [2] in the context of statistical inference. It was later developed by Shafer [3] into a general framework for uncertainty modeling. In the years since this earlier work, DS theory has been further developed and it has been identified as a framework for handling a wide variety of data imperfections, providing elegant solutions to real world problems [4], [5]. It has been transformed into an important computational tool for evidential reasoning in numerous application scenarios (e.g., expert systems) [6].

DS theory offers greater expressiveness and flexibility in evidential reasoning under uncertainty [7]. However, these advantages come at a cost: DS theoretic (DST) operations involve an additional cost in terms of higher computational complexity, especially when compared to methods based on the classical probability theory [8]. So, a major challenge for harnessing the advantages of DS theory in practice is to overcome this computational complexity, especially when working with large frames of discernment.

We do not believe that the present literature has given due attention to this issue [4], [6]. There is no widely accepted computationally feasible generalized framework to represent DST models and carry out DST operations. A thoughtful discussion about data structures and algorithms for efficient DST computations is still lacking. The development of an efficient computational framework is of critical importance if we are to harness the strengths of DS theory and make it more widely applicable in practice and in real-world scenarios.

DST implementations in current use are limited to computations on smaller frames of discernment, and they lack the ability to handle larger frames mainly due to the prohibitive computational burden they engender. A review of current implementations and applications [4], [6], [9] confirms that work is needed to overcome these computational limitations.

The work in this paper is an attempt to fill the void between what DS theory can offer and its practical implementation. For this purpose, we introduce a novel generalized computational framework where we develop three different representations — *DS-Vector, DS-Matrix,* and *DS-Tree* — which offer significantly greater computational capability for representation of DST models and DST operations. They also act as simple tools for visualization of DST models and the complex nature of the computations involved. A strategy, which we refer to as *REGAP (REcursive Generation of and Access to Propositions),* is developed and used in our development of this computational framework. Relevant data structures and generalized algorithms to work with the framework are discussed and compared in the paper.

Popular approaches to represent a focal element (i.e., a proposition which receives DST 'support') are the use of a bit-string [10]–[12] or an integer [13]. Binary representation has been used in combination, marginalization, and projection of multivariate belief functions [14], [15]. Binary representation has also been used to express the belief computation formulas using matrix calculus [16] and in fusion algorithm implementations [17]. In this paper, we introduce an implicit index calculation mechanism to represent a focal element, which reduces memory usage and significantly improves computational performance.

In order to address the high computational complexity of belief value (or belief potential) calculations, several approximation methods have also been developed. Most of these approximation algorithms provide lower bounds which are obtained by removing some of the focal elements with or without redistributing the corresponding belief potentials [18]–[22]. More sophisticated methods produce lower and upper bounds , thus improving the quality of the approximation [23], [24]. These methods provide approximate and not exact belief potentials. A fast Möbius transform, which is analogous to the fast Fourier transform (FFT), has also been developed toward efficient DST computations [10], [11], [25]. It is worth pointing out that Shafer has stated, *"It remains to be seen how useful the fast Möbius transform will be in practice. It is clear, however, that it is not enough to make arbitrary belief function computations feasible."* [7, p.348]. Using the REGAP strategy, we introduce a new approach to identify propositions that are relevant to a belief potential computation. This technique is useful for calculating arbitrary belief, plausibility, and commonality potentials from a minimum number of operations. Implementation of the relevant belief computation algorithms based on the proposed framework is discussed and compared with alternative approaches.

This paper is organized as follows: Section II provides a review of essential DST notions. Section III introduces our generalized computational framework. Section IV contains efficient algorithms for belief potential computation of arbitrary propositions. Section V contains comparisons and experimental results. Section VI provides the concluding remarks.

II. Basic Notions of DS Belief Theory

In DS theory, the *frame of discernment (FoD)* refers to the set of all possible mutually exclusive and exhaustive propositions [3]. We consider the case where the FoD is finite and we denote it as $\Theta = \{\theta_0, \theta_1, \dots, \theta_{n-1}\}$. Note that, for computational ease, we use the indices 0 and $n-1$ for the first and the last elements, respectively. Proposition θ_i, which is referred to as a *singleton,* represents the lowest level of discernible information. The power set of Θ, denotes by 2^Θ, form all the propositions of interest in DS theory. A proposition that is not a singleton is referred to as a *composite.* The cardinality of Θ is denoted by $|\Theta|$. The set $A \backslash B$ denotes all singletons in $A \subseteq \Theta$ that are not included in $B \subseteq \Theta$, i.e., $A \backslash B = \{\theta_i \in \Theta \mid \theta_i \in A, \theta_i \notin B\}$. We use $\overline{A}$ to denote $\Theta \backslash A$.

The most important representations used in DST representations and computations are the *basic belief assignment (BBA)* or *mass assignment* denoted by $m(\cdot)$, *belief* denoted by Bl, *plausibility* denoted by Pl, and *commonality* denoted by Q.

A. Basic Belief Assignment (BBA) or Mass Assignment

The mass represents the 'support' that is strictly allocated to a given proposition.

Definition 1 (Basic Belief Assignment (BBA))**.** *The mapping* $m : 2^\Theta \mapsto [0,1]$ *is said to be a* basic belief assignment *if*

$$m(\emptyset) = 0 \text{ and } \sum_{A \subseteq \Theta} m(A) = 1. \qquad \blacksquare$$

The masses in composite propositions are free to move into its individual singletons, which allows one to model the notion of *ignorance.* Complete ignorance can be modeled via the *vacuous BBA:* $m(A) = 1_\Theta \equiv 1$ for $A = \Theta$, and 0 for $A \subset \Theta$. Propositions that possess nonzero mass are referred to as *focal elements;* the set of all focal elements in a FoD is referred to as its *core* $\mathfrak{F}$, i.e., $\mathfrak{F} = \{A \subseteq \Theta \mid m(A) > 0\}$. Note that $|\mathfrak{F}|$ is the number of focal elements. $\mathcal{E} = \{\Theta, \mathfrak{F}, m(\cdot)\}$ is referred to as the *body of evidence (BoE).*

B. Belief

The belief assigned to a proposition takes into account the support for all of its subsets.

Definition 2 (Belief)**.** *Given a BoE* $\mathcal{E} = \{\Theta, \mathfrak{F}, m(\cdot)\}$*, the belief assigned to* $A \subseteq \Theta$ *is* $Bl : 2^\Theta \mapsto [0,1]$ *where*

$$Bl(A) = \sum_{B \subseteq A} m(B). \qquad \blacksquare$$

Propositions that posses nonzero belief are denoted by $\widehat{\mathfrak{F}}$, i.e., $\widehat{\mathfrak{F}} = \{A \subseteq \Theta \mid Bl(A) > 0\}$.

C. Plausibility

The plausibility measures the extent to which a proposition is plausible, i.e., the amount of belief not strictly supporting the complement of the proposition.

Definition 3 (Plausibility)**.** *Given a BoE* $\mathcal{E} = \{\Theta, \mathfrak{F}, m(\cdot)\}$*, the plausibility assigned to* $A \subseteq \Theta$ *is* $Pl : 2^\Theta \mapsto [0,1]$ *where*

$$Pl(A) = 1 - Bl(\overline{A}). \qquad \blacksquare$$

It is easy to see that

$$Pl(A) = \sum_{\substack{B \subseteq \Theta \\ B \cap A \neq \emptyset}} m(B). \qquad (1)$$

Moreover, $Pl(A) \geq Bl(A)$, $\forall A \subseteq \Theta$. The *uncertainty* $Un(A)$ associated with the proposition $A \subseteq \Theta$ is taken as the interval $Un(A) = [Bl(A), Pl(A)]$.

D. Commonality

The commonality quantifies the support for those propositions that imply a given proposition.

Definition 4 (Commonality)**.** *Given a BoE* $\mathcal{E} = \{\Theta, \mathfrak{F}, m(\cdot)\}$*, the commonality function of* $A \subseteq \Theta$ *is* $Q : 2^\Theta \mapsto [0,1]$ *where*

$$Q(A) = \sum_{A \subseteq B \subseteq \Theta} m(B). \qquad \blacksquare$$

III. A General Computational Framework

Henceforth, for convenience, we will use the notation N to denote 2^n, where $n = |\Theta|$. Note that, $N = 2^n$ is the maximum number of focal elements that a BoE could possess. Actually, the maximum number of focal elements is $N-1$, but this difference is immaterial especially when working with a large FoD. Also we will refer to the subsets of A and A itself as *subset propositions* of A and use the notation M to denote 2^m, where $m = |A|$.

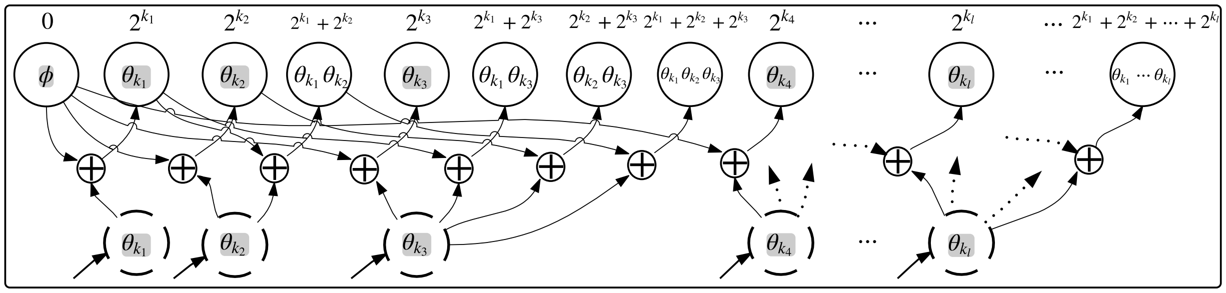
0
2^{k_1}
2^{k_2}
$2^{k_1}+2^{k_2}$
2^{k_3}
$2^{k_1}+2^{k_3}$
$2^{k_2}+2^{k_3}$
$2^{k_1}+2^{k_2}+2^{k_3}$
2^{k_4}
2^{k_l}
$2^{k_1}+2^{k_2}+\cdots+2^{k_l}$
ϕ
θ_{k_1}
θ_{k_2}
$\theta_{k_1}\theta_{k_2}$
θ_{k_3}
$\theta_{k_1}\theta_{k_3}$
$\theta_{k_2}\theta_{k_3}$
$\theta_{k_1}\theta_{k_2}\theta_{k_3}$
θ_{k_4}
θ_{k_l}
$\theta_{k_1}\cdots\theta_{k_l}$

Fig. 1. REGAP: REcursive Generation of and Access to Propositions.

A lookup table named $power$ is used to enhance the computational efficiency. It contains 2 to the power of singleton indexes in increasing order and was implemented using a dynamic array that replaces run-time computation of power values with a simpler array indexing operation. $Power[i]$ is referred to as 2 to the power of i, which is the i'th entry in the $power$ table.

Computational complexity of DST computations directly depends on the access speed of focal elements and corresponding belief values. In recent literature, the commonly used approach for such computations is the utilization of list structures for both focal elements and respective belief values, or hard coding relevant values [9], [13], [14]. Representation as a list or a set of pairs of focal elements and relevant belief values are also used. Accessing a focal element in list implementation requires cycling through the lists, and complexity of this operation is $\mathcal{O}(|\mathfrak{F}|)$. In the novel generalized computational framework that is being proposed in this paper, propositions are identified via an implicit index. Therefore, there is no overhead on storing focal elements as a bit-string, an integer, or any other method; only belief potentials require to be stored. Due to the efficient access operations, the proposed method can be utilized to gain a significant advantage on computational efficiency when dealing with large BoEs. List structures become attractive when considering the memory usage aspect of static BoEs, with a fewer number of focal elements.

The proposed belief computation strategy provides generated indices. Therefore, computational complexity of an element access is $\mathcal{O}(1)$ (i.e., constant) during belief computations. In the provided general access algorithms which include index generation, the computational complexity is $\mathcal{O}(m)$.

A. REGAP: REcursive Generation of and Access to Propositions

Consider the FoD $\Theta = \{\theta_0, \theta_1, \ldots, \theta_{n-1}\}$. Suppose we desire to determine the belief potential associated with $A = (\theta_{k_1}, \theta_{k_2}, \ldots, \theta_{k_\ell}\} \subseteq \Theta$. The REGAP property allows us to recursively generate the propositions that are relevant for this computation: Start with $\emptyset$. First insert the singleton $\theta_{k_1} \in A$. Only one proposition is associated with this singleton, viz., $\emptyset \cup \theta_{k_1} = \theta_{k_1}$ itself. Next insert another singleton $\theta_{k_2} \in A$. The new propositions that are associated with this singleton are $\emptyset \cup \theta_{k_2} = \theta_{k_2}$ and $\theta_{k_1} \cup \theta_{k_2} = (\theta_{k_1}, \theta_{k_2})$. Inserting another singleton $\theta_{k_3} \in A$ brings the new propositions $\emptyset \cup \theta_{k_3} = \theta_{k_3}$, $\theta_{k_1} \cup \theta_{k_3} = (\theta_{k_1}, \theta_{k_3})$, $\theta_{k_2} \cup \theta_{k_3} = (\theta_{k_2}, \theta_{k_3})$, and $(\theta_{k_1}, \theta_{k_2}) \cup \theta_{k_3} = (\theta_{k_1}, \theta_{k_2}, \theta_{k_3})$. In essence, when a new singleton is added, new propositions associated with it can be recursively generated by adding the new singleton to each existing proposition. Of course, all propositions of interest within the FoD Θ can be generated when $A = \Theta$.

We refer to this recursive scheme as *REGAP,* which stands for *REcursive Generation of and Access to Propositions*. It is illustrated in Fig. 1. These recursively generated propositions can be formulated as a vector, a matrix or a tree, and utilized to represent a dynamic BoE.

1) DS-Vector: Vector Representation of a Dynamic BoE: All propositions generated via REGAP can be represented using a dynamic vector, which we refer to as a *DS-Vector.* This is illustrated in Fig. 2. It can also be used as a visualization tool of a dynamic BoE.

From a computational point of view, a DS-Vector can be viewed as a dynamic array data structure [26]. Propositions are represented by implicit contiguous indexes, which are considered as implicit bit-strings or decimal integers. So, no memory allocation is needed to store a proposition. Memory allocation is needed only to store the required belief potentials (or, more generally, mass, belief, plausibility, or commonality potentials).

Algorithm 1 provides an implementation to access a belief potential in $\mathcal{O}(m)$ complexity. When the proposition index is available, this becomes a constant time (i.e., $\mathcal{O}(1)$) operation.

Algorithm 1 Access a belief potential in a DS-Vector
```
1: procedure AccessPotential(Singletons $A$)
2:     $index \leftarrow 0$
3:     for each $\theta_i$ in $A$ do
4:         $index \leftarrow index + power[i]$
5:     end for
6:     Return $potential[index]$
7: end procedure
```

The salient steps in the algorithm are as follows:

Line #1: The required proposition can be passed as a bit-string or an integer. If so, this segment has to be replaced

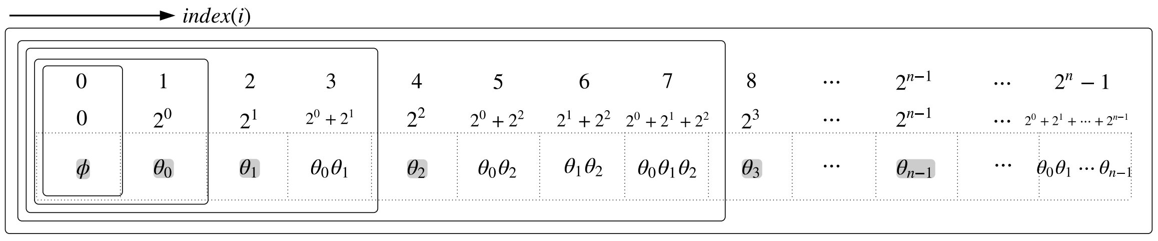

Fig. 2. **DS-Vector:** Vector representation of a dynamic BoE.

by a single input parameter (a bit-string or an integer). In this algorithm, the input parameter is represented in a more general way and passed as A, which includes all the constituent singleton propositions of the proposition of interest A.

Lines #3-5: Implicit index of the proposition is calculated by adding 2 to the power of indexes ($power[i]$) relevant to all the singleton propositions in A.

Line #6: Propositions are represented by implicit indexes and belief potentials are stored in the contiguous memory segments of the DS-Vector. Thus, $potential[index]$ retrieves the respective belief potential.

2) DS-Matrix: Matrix Representation of a Dynamic BoE: All propositions generated via REGAP can also be represented using a dynamic matrix as illustrated in Fig. 3. We refer to this as a *DS-Matrix.*

From a computational point of view, the DS-Matrix can be implemented using a dynamic array of dynamic arrays [26]. Propositions are represented by two implicit contiguous indexes (i and j), which are considered as two implicit bit-strings or two decimal integers.

Algorithm 2 provides an implementation to access a belief potential in $\mathcal{O}(m)$ complexity.

Algorithm 2 Access a belief potential in a DS-Matrix

```
 1: procedure ACCESSPOTENTIAL(EvenSingletons $A_e$, OddSingletons $A_o$)
 2:     $row \leftarrow 0$
 3:     $col \leftarrow 0$
 4:     for each $\theta_i$ in $A_o$ do
 5:         $row = row + power[i]$
 6:     end for
 7:     for each $\theta_i$ in $A_e$ do
 8:         $col = col + power[i]$
 9:     end for
10:     Return $potential[row][col]$
11: end procedure
```

The main steps in the algorithm are as follows:

Line #1: Input parameters are passed as A_e and A_o; A_e includes even numbered singletons and A_o includes odd singletons of the proposition of interest A ($A = A_e \cup A_o$).

Lines #4-6: Row index is computed by adding 2 to the power of existing odd singleton indexes ($power[i]$) in A_o.

Lines #7-9: Column index is calculated from the addition of 2 to the power of even singleton indexes ($power[i]$) in A_e.

Line #10: Belief potentials can be accessed by implicit row index and implicit column index; $potential[row][col]$ retrieves the respective belief potential.

3) DS-Tree: Perfectly Balanced Binary Tree Representation of a Dynamic BoE: Another way to represent all propositions generated via REGAP is a perfectly balanced binary tree [26] as illustrated in Fig. 4. We refer to this as a *DS-Tree.* When a new singleton is added, the singleton proposition itself stays as the root. Previous propositions stay on the left sub-tree. The new propositions relevant to the incoming singleton are generated by applying REGAP. Those elements stay on the right sub-tree.

Propositions are represented by relative positions according to the illustration provided in Fig. 4. So focal elements can be interpreted as implicit bit-strings or decimal integers. Thus, no memory allocation is needed to store propositions. Only respective belief potentials are contained in the nodes.

Representation of propositions follow the perfectly balanced binary search tree properties [27]. Therefore, it can be implemented using a dynamic array and follows DS-Vector properties. When the DS-Tree complies with classical binary tree implementations, Algorithm 3 provides access to a belief potential in $\mathcal{O}(log(N))$ (or $\mathcal{O}(n)$).

A belief potential of a proposition can be accessed by traversing through the implicit indexes of the binary tree. The important steps in the algorithm are as follows:

Line #1: Input parameters are the FoD Θ, constituent singleton propositions of the proposition of interest A, and the DS-Tree T.

Lines #5-7: Index of the proposition is computed by adding 2 to the power of singleton indexes ($power[i]$) in A.

Line #8: The binary tree is traversed down, starting from the root node, until the condition $index \bmod power[level] > 0$ is satisfied.

Lines #10-11: Left sub-tree is traversed if the $index$ is less than the current implicit index.

Lines #12-18: Right sub-tree is traversed if the $index$ is greater than the current implicit index.

Line #21: Required $node$ relevant to the proposition is obtained at end of the traversal. So $node.potential$ gives the relevant belief potential.

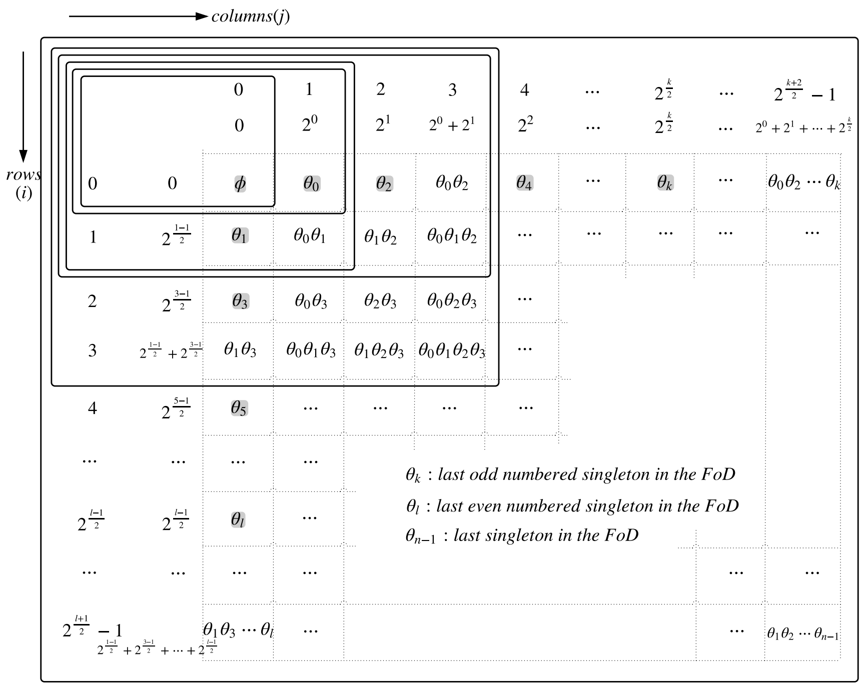
columns(j)
rows (i)
θ_k : last odd numbered singleton in the FoD
θ_l : last even numbered singleton in the FoD
θ_{n-1} : last singleton in the FoD

Fig. 3. **DS-Matrix:** Matrix representation of a dynamic BoE.

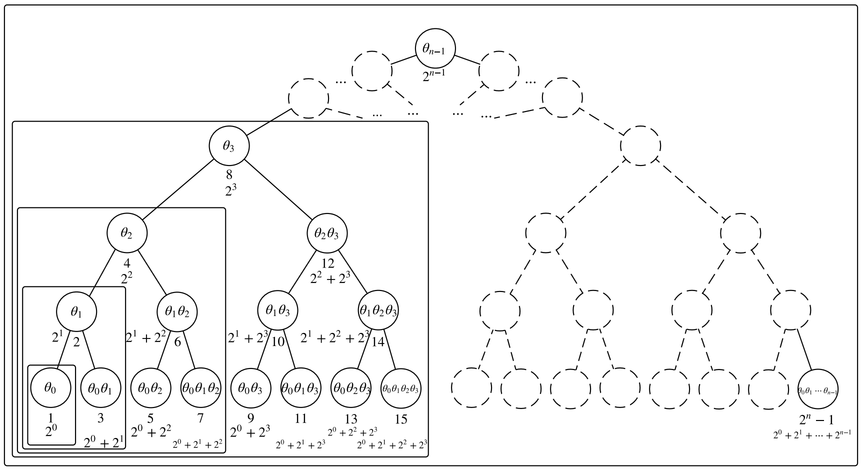

Fig. 4. **DS-Tree:** Perfectly balanced binary tree representation of a dynamic BoE.

IV. Efficient Algorithms for Arbitrary Belief Computations

Fast Möbius transform was developed towards addressing the high computational complexity of belief potential calculations [10], [11], [25]. However, it is inadequate to make arbitrary belief function computations feasible, especially when working with large FoDs [7]. Employing REGAP, we propose a new approach to identify propositions relevant to a given belief computation. This technique can be used to calculate arbitrary belief, plausibility, and commonality function values from a minimum number of operations.

A. Belief Calculation

List structure implementation is the commonly utilized approach to store mass potentials [9], [12]–[14]. Belief calculation requires cycling through the list structure to recognize whether each focal element should be included in the computation. Thus, computational complexity of this operation is $\mathcal{O}(|\mathfrak{F}|)$.

The REGAP strategy offers an alternative to generate the required propositions relevant to the computation of $Bl(A)$, where $A \subseteq \Theta$. In this method, belief computation is performed by accessing only the subset propositions. The maximum number of subset propositions that one would have to access is about $M = 2^m$, where $m = |A|$.

Algorithm 3 Access a belief potential in a DS-Tree

```
procedure ACCESSPOTENTIAL(FoD $\Theta$, Singletons $A$, DS-Tree $T$)
    $index \leftarrow 0$
    $level \leftarrow |\Theta| - 1$
    $node \leftarrow T.root$
    for each $\theta_i$ in $A$ do
        $index \leftarrow index + power[i]$
    end for
    $temp \leftarrow index$
    while $temp$ mod $power[level] > 0$ do
        if $temp/power[level] = 0$ then
            $node \leftarrow node.left$
        else if $temp/power[level] = 1$ then
            $temp \leftarrow temp - power[level]$
            if $temp = 0$ then
                break the loop
            end if
            $node \leftarrow node.right$
        end if
        $level \leftarrow level - 1$
    end while
    Return $node.potential$
end procedure
```

1) DS-Vector: Algorithm 4 provides an implementation to compute a belief potential in $\mathcal{O}(M)$ (or $\mathcal{O}(2^m)$) complexity.

Algorithm 4 Computing Belief in a DS-Vector

```
procedure COMPUTEBELIEF(Singletons $A$, Normalize $Nlz$)
    $belief \leftarrow 0$
    $count \leftarrow 0$
    for each $\theta_i$ in $A$ do
        $index[count] \leftarrow power[i]$
        $temp \leftarrow count$
        $count \leftarrow count + 1$
        for $j \leftarrow 0,\ temp - 1$ do
            $index[count] \leftarrow index[j] + power[i]$
            $count \leftarrow count + 1$
        end for
    end for
    for $i \leftarrow 0,\ power[|A|] - 2$ do
        $belief \leftarrow belief + potential[index[i]]$
    end for
    Return $belief/Nlz$
end procedure
```

The main steps in the algorithm are as follows:

Line #1: Input parameters are A and normalizing constant Nlz. Normalizing constant is the summation of all the mass potential values. mass potential are stored in raw values to improve the performance.

Lines #4-12: Subset propositions of A are generated by applying REGAP.

Lines #13-15: Belief is the summation of the potentials of relevant generated implicit indexes. Computational complexity of an iteration is $\mathcal{O}(1)$ (access operation).

Line #16: Normalized belief potential is the output parameter of the procedure.

2) DS-Matrix: Algorithm 5 can be used to compute a belief potential in $\mathcal{O}(M)$ (or $\mathcal{O}(2^m)$) complexity.

Algorithm 5 Computing Belief in a DS-Matrix

```
procedure COMPUTEBELIEF(SingletonCoordinates $A_P$, Normalize $Nlz$)
    $belief \leftarrow 0$
    $count \leftarrow 0$
    for each pair $p$ in $A_P$ do
        $index[count].row \leftarrow p.row$
        $index[count].col \leftarrow p.col$
        $temp \leftarrow count$
        $count \leftarrow count + 1$
        for $j \leftarrow 0, temp - 1$ do
            $index[count].row \leftarrow index[j].row + p.row$
            $index[count].col \leftarrow index[j].col + p.col$
            $count \leftarrow count + 1$
        end for
    end for
    for $i \leftarrow 0,\ power[|A_P|] - 2$ do
        $belief \leftarrow belief$
        $+potential[index[i].row][index[i].col]$
    end for
    Return $belief/Nlz$
end procedure
```

The salient steps in the algorithm are as follows:

Line #1: Input parameters are A_P and normalizing constant Nlz. A_P contains row and column coordinate pairs of all the singleton propositions of the proposition of interest A.

Lines #4-14: Subset propositions of A are obtained by applying REGAP.

Lines #15-18: Belief is the summation of the potentials relevant to the generated implicit index pairs. Computational complexity of an iteration is $\mathcal{O}(1)$ (access operation).

Line #19: Procedure returns the normalized belief potential.

3) DS-Tree: Algorithm 6 can be used to compute a belief potential in $\mathcal{O}(M\log(N))$ (or $\mathcal{O}(2^m n)$) complexity when the DS-Tree is implemented using node structures. Dynamic array implementation of the DS-Tree complies with algorithm 4 and the belief computation complexity is $\mathcal{O}(M)$ (or $\mathcal{O}(2^m)$).

The salient steps in the algorithm are as follows:

Line #1: Input parameters are the proposition of interest A, DSTree T, and normalizing constant Nlz.

Lines #4-12: Subset propositions of A obtained by applying REGAP.

Lines #13-30: Belief is the summation of the potentials of generated implicit indexes. Computational complexity of a tree traversal iteration is $\mathcal{O}(\log(N))$ (or $\mathcal{O}(n)$) and follows the Algorithm 3.

Algorithm 6 Computing Belief in a DS-Tree

```
1: procedure ComputeBelief(Singletons $A$, DSTree $T$, Normalize $Nlz$)
2:     $belief \leftarrow 0$
3:     $count \leftarrow 0$
4:     for each $\theta_i$ in $A$ do
5:         $index[count] \leftarrow power[i]$
6:         $temp \leftarrow count$
7:         $count \leftarrow count + 1$
8:         for $j \leftarrow 0$, $temp - 1$ do
9:             $index[count] \leftarrow index[j] + power[i]$
10:            $count \leftarrow count + 1$
11:        end for
12:    end for
13:    for $i \leftarrow 0$, $power[|A|] - 2$ do
14:        $level \leftarrow |\Theta| - 1$
15:        $leaf \leftarrow T.root$
16:        $index \leftarrow index[i]$
17:        while $index \bmod power[level] > 0$ do
18:            if $index/power[level] = 0$ then
19:                $leaf \leftarrow leaf.left$
20:            else if $index/power[level] = 1$ then
21:                $index \leftarrow index - power[level]$
22:                if $index = 0$ then
23:                    break the loop
24:                end if
25:                $leaf \leftarrow leaf.right$
26:            end if
27:            $level \leftarrow level - 1$
28:        end while
29:        $belief \leftarrow belief + leaf.mass$
30:    end for
31:    Return $belief/Nlz$
32: end procedure
```

Line #31: Normalized belief potential is the output parameter of the procedure.

B. Plausibility and Commonality Calculation

Plausibility $Pl(A)$ can be computed by observing that $Pl(A) = 1 - Bl(\overline{A})$ and applying a belief computation algorithm to $\overline{A}$. Propositions relevant to commonality $Q(A)$ calculation can be generated by applying REGAP to $\overline{A}$ and adding proposition A to all the generated propositions. In this way, computations can also be performed with minor modifications to Algorithms 4, 5, and 6. Computational complexity remains the same as for belief computation algorithms.

V. Experiments

We have developed a novel belief computation library [28] using the C++ programming language. This includes the implementation of data structures and algorithms for a generalized computational framework, in particular, the three representations *DS-Vector, DS-Matrix,* and *DS-Tree,* for representing DST models and carrying out DST operations. All experiments were simulated on a Macintosh desktop computer (iMac) running Mac OS X 10.11.3, with 2.9GHz Intel Core i5 processor and 8GB of 1600MHz DDR3 RAM.

Average computational times for accessing arbitrary propositions are listed in Table I for different implementations: DS-Vector in algorithm 1, DS-Matrix in algorithm 2, and common list structure implementation. Results were obtained by executing the algorithms for 100000 randomly chosen propositions from the FoD and noting the average CPU time. A random set of focal elements were generated in the core for each FoD size.

FoD Size	Max. $\lvert\mathfrak{F}\rvert$	DS-Vector	DS-Matrix	List Struct.
2	3	0.379	0.393	0.465
4	15	0.400	0.412	0.510
6	63	0.410	0.454	0.739
8	255	0.443	0.449	1.541
10	1023	0.433	0.496	4.632
12	4095	0.465	0.493	16.906
14	16383	0.465	0.527	67.242
16	65535	0.495	0.517	268.443
18	262143	0.529	0.560	1124.060
20	1048575	0.575	0.629	4609.370

TABLE I
CPU Time of Accessing a Proposition (μs)

From Table I, the speed advantage of the three proposed implementations over the commonly used list structure implementation is quite clear. With increasing FoD size, the access times for DS-Vector and DS-Matrix almost remain the same when compared to the rapid growth of the computational times in list structure implementation.

Average computational times of a randomly chosen belief computation using the algorithm 4, algorithm 5, and list structures are given in Table II. Results were obtained by executing the algorithms for 100000 randomly chosen propositions and noting the average CPU time. A random set of focal elements were generated in the core for each FoD size.

FoD Size	Max. $\lvert\mathfrak{F}\rvert$	DS-Vector	DS-Matrix	List Struct.
2	3	0.373	0.362	0.450
4	15	0.378	0.376	0.531
6	63	0.415	0.450	0.833
8	255	0.453	0.508	1.779
10	1023	0.525	0.663	5.529
12	4095	0.655	0.923	20.757
14	16383	0.884	1.314	81.196
16	65535	1.340	2.159	325.930
18	262143	2.107	3.510	1373.110
20	1048575	3.963	6.210	5448.170

TABLE II
CPU Time of a Belief Computation (μs)

The proposed implementations offer better results when compared with the commonly used list structure implementation. With increasing FoD size, the average computational time of a randomly chosen belief function increases very slowly with DS-Vector and DS-Matrix in comparison to the list structure implementation. The DS-Tree provides the same experimental results as DS-Vector.

VI. Concluding Remarks

.

This paper provides a general framework along with data structures and efficient algorithms for DST computations. The data structures presented can also serve as tools for BoE visualization. We believe that the proposed implementations constitute a significant step forward in harnessing the strengths of DS theory in practical application scenarios.

The implicit index calculation mechanism that we introduce for the purpose of representing a proposition serves to reduce the memory usage and to significantly improve computational efficiency. All the indexes are calculated according to relative positions in the data structures. Only belief potentials need to be stored. Memory usage efficiency is greatly improved since representing the proposition as a bit-string or an integer is unnecessary.

When the index of the proposition is available, the proposed algorithms for accessing a proposition take a constant time irrespective of the FoD's size. Therefore updating a belief potential (or a mass potential) in a BoE can be executed in significantly less time. The proposed REGAP strategy is invaluable in that it allows one to identify the exact subsets relevant to a given belief computation. Efficient algorithms for belief calculation of arbitrary propositions are also developed by ensuring that only a minimum possible number of operations are executed.

Representation of DS-Tree propositions follow the perfectly balanced binary search tree properties. Therefore, it can be implemented using a dynamic array and can gain the performance relevant to a DS-Vector. As an outcome of this research work, a belief computation library in C++ which includes all the important operations to work with belief computations is made available [28]. This may be useful for a significant performance improvement for applications based on DST methods.

Our current research explores at how to incorporate the ability to add and remove singletons from the FoD, i.e., how to handle dynamic FoDs. Removing one singleton from a FoD removes half the propositions that need to be considered. Thus, from a computational perspective, the ability to add, remove, and change the FoD is highly important. Based on the proposed computational framework, it is also possible to develop efficient algorithms for Demspter's combination rule, DST conditional [29], the conditional update equation [30], and other operations associated with DST algorithms.

Acknowledgment

This work is based on research supported by the U.S. Office of Naval Research (ONR) via grant #N00014-10-1-0140, and the U.S. National Science Foundation (NSF) via grant #1343430.

References

[1] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *Ann. Math. Stat.*, vol. 38, no. 2, pp. 325–339, 1967.

[2] ——, "A generalization of bayesian inference," *J. Roy. Statistical Soc.*, vol. 30, no. 2, pp. 205–247, 1968.

[3] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.

[4] R. R. Yager, J. Kacprzyk, and M. Fedrizzi, Eds., *Advances in the Dempster-Shafer Theory of Evidence*. New York: Wiley, 1994.

[5] P. Smets, "Practical uses of belief functions," in *Proc. Conf. 15th Uncertainty in Artificial Intelligence (UAI)*, K. B. Laskey and H. Prade, Eds. San Francisco, CA: Morgan Kaufmann, 1999, pp. 612–621.

[6] R. R. Yager and L. Liu, Eds., *Classic Works of the Dempster-Shafer Theory of Belief Functions*. Heidelberg, Germany: Springer-Verlag, 2008.

[7] G. Shafer, "Perspectives on the theory and practice of belief functions," *Int. J. Approx. Reasoning*, vol. 4, no. 5-6, pp. 323–362, Oct. 1990.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann, 1988.

[9] SIPTA. (2016) Software tools for imprecise probabilities. [Online]. Available: http://www.sipta.org/index.php?id=sfw

[10] H. M. Thoma, "Factorization of belief functions," Ph.D. dissertation, Harvard University, Cambridge, MA, 1989.

[11] ——, "Belief function computations," in *Conditional Logic in Expert Systems*, I. R. Goodman, M. M. Gupta, H. T. Nguyen, and G. S. Rogers, Eds. Amsterdam: North-Holland, 1991, pp. 269–308.

[12] H. Xu and R. Kennes, "Steps toward efficient implementation on dempster-shafer theory," in *Advances in the Dempster-Shafer Theory of Evidence*, R. R. Yager, J. Kacprzyk, and M. Fedrizzi, Eds. New York: Wiley, 1994, pp. 153–174.

[13] L. Liu, "A relational representation of belief functions," in *Belief Functions: Theory and Applications*, F. Cuzzolin, Ed. Switzerland: Springer, 2014, pp. 161–170.

[14] R. Haenni and N. Lehmann, "Implementing belief function computations," *Int. J. Intelligent Syst.*, vol. 18, no. 1, pp. 31–49, 2003.

[15] N. Lehmann, "Fast projection of focal sets," in *Proc. 3rd Int. Conf. Intelligent Sensors, Sensor Networks and Information*, Melbourne, Australia, Dec 2007, pp. 191–196.

[16] P. Smets, "The application of the matrix calculus to belief functions," *Int. J. Approx. Reasoning*, vol. 31, no. 1, pp. 1–30, 2002.

[17] H. Wu, "Sensor data fusion for context-aware computing using dempster-shafer theory," Ph.D. dissertation, Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, 2003.

[18] F. Voorbraak, "A computationally efficient approximation of dempster-shafer theory," *Int. J. Man-Machine Stud.*, vol. 30, no. 5, pp. 525–536, 1989.

[19] D. Dubois and H. Prade, "Consonant approximations of belief functions," *Int. J. Approx. Reasoning*, vol. 4, no. 5, pp. 419–449, 1990.

[20] B. Tessem, "Approximations for efficient computation in the theory of evidence," *Artificial Intell.*, vol. 61, no. 2, pp. 315–329, 1993.

[21] M. Bauer, "Approximation algorithms and decision making in the dempster-shafer theory of evidencean empirical study," *Int. J. Approx. Reasoning*, vol. 17, no. 2, pp. 217–237, 1997.

[22] D. Harmanec, "Faithful approximations of belief functions," in *Proc. Conf. 15th Uncertainty in Artificial Intelligence (UAI)*. San Francisco, CA: Morgan Kaufmann, 1999, pp. 271–278.

[23] T. Denœux, "Inner and outer approximation of belief structures using a hierarchical clustering approach," *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 9, no. 4, pp. 437–460, 2001.

[24] R. Haenni and N. Lehmann, "Resource bounded and anytime approximation of belief function computations," *Int. J. Approx. Reasoning*, vol. 31, no. 1, pp. 103–154, 2002.

[25] R. Kennes and P. Smets, "Fast algorithms for dempster-shafer theory," in *Uncertainty in Knowledge Bases*, B. Bouchon-Meunier, R. R. Yager, and L. A. Zadeh, Eds. Berlin: Springer-Verlag, 1990, pp. 14–23.

[26] R. Sedgewick, *Algorithms in C++*, 3rd ed. Boston, MA: Addison-Wesley, 2002.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: The MIT Press, 2009.

[28] ProFuSELab. (2016) Belief computation library. [Online]. Available: https://github.com/ProFuSELab/Belief-Computation-Library

[29] R. Fagin and J. Y. Halpern, "A new approach to updating beliefs," in *Proc. Conf. 6th Uncertainty in Artificial Intelligence (UAI)*, P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, Eds. New York: Elsevier Science, 1991, pp. 347–374.

[30] T. L. Wickramarathne, K. Premaratne, M. N. Murthi, M. Scheutz, S. Kuebler, and M. Pravia, "Belief theoretic methods for soft and hard data fusion," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 2388–2391.