Map-reduce Implementation of Belief Combination Rules

Frédéric Dambreville
Ensta Bretagne, 2 rue de Verny, Brest, France
DGA MI, Bruz, France

Keywords: Belief Functions, Combination Rules, Statistic, Map-reduce.

Abstract: This paper presents a generic and versatile approach for implementing combining rules on preprocessed belief functions, issuing from a large population of information sources. In this paper, we address two issues, which are the intrinsic complexity of the rules processing, and the possible large amount of requested combinations. We present a fully distributed approach, based on a map-reduce (Spark) implementation.

1 INTRODUCTION

Thhis paper addresses the issue of generic computation of belief combinations in the context of a large-scale networked community of agents (*eg.* in a social network) or sensored sources (*eg.* surveillance camera). These agents or sensors produce information, which are preprocessed under the form of belief functions assigned to representative propositions. A preprocessing of the networked agents/sensors and of the produced information then infers a large collection of belief function clusters, which are representative of agent/sensors viewpoints on a common topic. The combination of belief functions of a cluster by means of dedicated combination rules infers a refined analysis of the relative viewpoints, including agreement and disagreement, on the considered topic.

Emerging works (Zhou et al., 2015a) have been done on the application of belief function to the analysis of interaction between agents of a social network on the basis of shared semantic content. These works are especially based (Zhou et al., 2015b) on evidential clustering of agents resulting in a fuzzy identification of communities. These clustering algorithms optimize the evidential similarities/disimilarities between agents, but do not deeply involve combinations of beliefs and the semantic it could extract. Of course, an issue of belief functions application, especially when sources are combined, is the efficient computation of many cases of combination in order to evaluate each connection between networked agents.

In the domain of surveillance, (Liu et al., 2009) have underlined eight challenge of a video network. Among them are the uncertainty of events, inconsistency or conflict between multiple sources, the composition of elemental events, and the scalability of the system. (Hong et al., 2014) have applied evidential networks to the problem of video surveillance in a controlled application (Smart transport). The struture implied by the networks makes possible an efficient reduction of the computational complexity.

When dealing with unstructured network, computational approaches for computing raw combinations in numbers is a necessary tool. Now, the issue of the generated conflict has been challenged by many evolutions of the historical conjunctive and Dempster-Shafer rules (Dubois and Prade, 1986; Lefevre et al., 2002; Smarandache and Dezert, 2005; Florea et al., 2006; Martin and Osswald, 2007). The development of generic implementation (Dambreville, 2009) of combination rules is a challenge in itself. Our work consider both issues by extending a previous work (Dambreville, 2009) dedicated to the generic implementation of rules. In the continuation of this work, this paper considers the problem of parallel and pooling computation by factoring the combination process, and a map-reduce (Dean and Ghemawat, 2008) approach is proposed within the framework Apache SPARK (Zaharia et al., 2010). In order to factor some combination rules, new algebraic structures (*eg.* multisets in the case of Dubois&Prade rule) are used as a processing space instead of the propositional framework of belief functions.

Section 2 introduces basic concept on belief functions. Section 3 presents our new contribution for a parallel and pooling implementation of combination rules and our previous work (Dambreville, 2009) is also introduced. Section 4 presents some limited tests.

Dambreville, F.
Map-reduce Implementation of Belief Combination Rules.
DOI: 10.5220/0005987001440149
In *Proceedings of the 5th International Conference on Data Management Technologies and Applications (DATA 2016)*, pages 144-149
ISBN: 978-989-758-193-9

2 BELIEF FUNCTIONS

Belief functions are representations of imprecise and uncertain information over an algebraic framework, a lattice in its most general form. Most authors consider belief functions over powersets, as a Boolean algebra, and this is our hypothesis here. From now is given the finite set Ω, the universe or frame of discernment.

2.1 Belief Assignments

The imprecise and uncertain information are characterized by basic belief assignment (bba), m, over propositions of the framework. Thus, a bba is defined as the attribution of pieces of belief to subsets of Ω:

$$m \geq 0 \text{ and } \sum_{X \subseteq \Omega} m(X) = 1 \,.$$

In case of *closed world hypothesis*, it is assumed that the belief put on empty set is zeroed, *ie.* $m(\emptyset) = 0$. This paper does not discuss further about such hypothesis, but its involvement does not imply any difficult generalization. However, we refer subsequently to $m(\emptyset)$ as the conflict related to m, and consider rules which are based on a redistribution of the conflict.

Given M sources providing information by means of bba $m_{1:M}$, the fusion of these information are computed by combinations rules. In the case of a closed world, the classical combination rule of Dempster-Shafer (Dempster, 1968; Shafer, 1976) is derived from the conjunctive rule by means of a normalization based on the conflict. Without lost of generality (Smets, 1990), Dempster-Shafer rules could be rewritten as a conjunctive rule without normalization in the case of open world.

2.2 Combination Rules

Given two sources of information, characterized by their respective bbas, m_1, m_2, the conjunctive combination of m_1 and m_2 is defined by:

$$m_1 \circledcirc m_2(X) \quad = \sum_{Y_1, Y_2 : Y_1 \cap Y_2 = X} m_1(Y_1) m_2(Y_2) \,.$$

The conjunctive rule only works for open world hypothesis, since it is possible to have $m_1 \circledcirc m_2(\emptyset) > 0$ while $m_1(\emptyset) = m_2(\emptyset) = 0$.

By interpreting $m_1 \circledcirc m_2(\emptyset)$ as a measure of conflict and redistributing it, many alternative rules have been proposed. As example, Dubois & Prade rule (Dubois and Prade, 1986) and PCR5/PCR6 rule (Smarandache and Dezert, 2005; Martin and Osswald, 2007) redistribute the conflict with different approaches:

- **Dubois & Prade Rule.** The rule proposed by Dubois and Prade extends the conjunctive rule by redistributing disjunctively the conflict. Appendix 5 presents its original definition,
- **PCR Rules.** The PCR combination rules, pioneered by Dezert and Smarandache (Smarandache and Dezert, 2005), are based on a local proportional redistribution of the conflict.

As an example of implementation, we consider specifically Dubois & Prade rule, generalized to many sources, but the approach is generic and addresses a potentially large scope of rules.

3 IMPLEMENTATIONS

It is assumed that the bba $m_{1:M}$, to be fused, are taken amongst a set of bba $\widetilde{m}_{1:N}$, where $2 \leq M \ll N$. Typically, $m_i = \widetilde{m}_{j[i]}$, where the *selection* map $j \in \{1:N\}^{\{1:M\}}$ is injective in general. The combination of $m_{1:M}$ is done for selection j. The number of concerned selections could be very large. Our main concern and challenge is to implement the computation of combination rules for all selections as a distributed process. A *map-reduce* approach (Dean and Ghemawat, 2008) is considered for this computation. A first approach is based on a previous work.

Section 3.1 introduces the generic formalism of referee functions (Dambreville, 2009) for defining combination rules. On this basis, section 3.2 presents a map-reduce implementation of the combination rules. Section 3.3 enhances the formalism of referee functions with Markov properties, and improves the definition of the combination rules, with recursive computational properties. On this basis, section 3.4 presents a map-reduce and recursive implementation.

Notation: Indicator functions are defined by:

$$I[P] = \begin{cases} 0 \text{ if } P \text{ is } \textbf{false} \\ 1 \text{ if } P \text{ is } \textbf{true} \end{cases} \tag{1}$$

3.1 Formulation based on Indicators

(Dambreville, 2009) proposed a generic formulation of combination rules by means of conditional functions (*referee functions*), denoted $F(X|Y_{1:M}; m_{1:M})$, which have a computational meaning as indicator functions. In this framework, the combinations of bba $m_{1:M}$ is expressed under the form:

$$\oplus[m_{1:M}|F](X) = \sum_{Y_{1:M} \subseteq \Omega} F(X|Y_{1:M}; m_{1:M}) \prod_{i=1}^{M} m_i(Y_i) \,. \tag{2}$$

In this formulation, a distinction is made between the rule processing expressed by the summation, and the rule definition which is expressed by the conditional indicator F. It is easy, then, to imply a generic distributed implementation of this summation, and we propose an implementation within Spark framework (Zaharia et al., 2010). This generic definition by means of indicator function is quite general however, as shown in (Dambreville, 2009), and typically, there are referee functions defined for conjunctive or disjunctive rules, D&P rules, PCR6 rule, and more. For the concern of this paper, we present only the referee function related to D&B rules.

3.1.1 Alternative Definition of D&P Rule

The rule of Dubois and Prade (Dubois and Prade, 1986) is naturally generalized to more than two sources by redistributing the conflict on the disjunction of the best consensuses:

$$m_1 \oplus_{DP} \cdots \oplus_{DP} m_M = \oplus \left[m_{1:M} | F_{DP} \right] , \qquad (3)$$

where:

$$F_{DP}(X|Y_{1:M};m_{1:M}) = I\left[X = \arg\max_{\omega\in\Omega} \sum_{i=1}^{M} I\left[\omega \in Y_i\right] \right] . \qquad (4)$$

Set $\arg\max_{\omega\in\Omega} \sum_{i=1}^{M} I\left[\omega \in Y_i\right]$ is the subset of Ω, whose elements receive the best vote derived from their belonging to propositions $Y_{1:M}$.

3.1.2 Computational Issues

There are actually two aspects to be considered, since the computation may be computing-intensive as well as data-intensive. On the one hand, it may be computing intensive, since a belief assignment is a vector of dimension $2^{\mathrm{card}(\Omega)}$; without approximation, the complexity of any belief computation increases dramatically with the size of the frame of discernment, and this issue is worsened with the number of bba to be combined. As a perspective of a distributed intensive computation of the rules, is the possibility to handle complex belief representations and their combinations for specific applicative use or conceptual studies. On the other hand, the computation may be data intensive, in the case of multiple combinations among a large collection of bba, typically issuing from local processing related to a collection of sources of information. In this kind of application, the many sources of information produce pieces of data, from which knowledges are extracted by the local process in the form of bba in the context of a given logical frame. Then, the extracted bba are combined according to a combination plan, caracterized by a selection function, in order to evaluate the compatibility of the sources or evaluate a confirmed knowledge. The combination plan generally implies a large amount of combination cases. Although our approach may be applied to both case, our preliminary and limited tests focus on the second scenario. Now, we do not address here the question of the extraction, but only the question of the combination.

3.2 Map-reduce Implementation

We implemented the generic fusion process (2) by means of a Map-reduce principle (Dean and Ghemawat, 2008). This implementation has been made by means of SPARK (Zaharia et al., 2010) Resilient Distributed Dataset (RDD) with the following steps:

Mapping steps: In these steps, the inner computations are done, that is the joint belief assignments, $\prod_{i=1}^{M} m_i(Y_i)$, and the definition maps, $F(X|Y_{1:M};m_{1:M})$. These computations are derived for all considered selection maps j and all possible non-zero propositional combinations, $Y_{1:M}$. The amount of data is potentially exponential with M.

- Define the set of *selection* maps $j \in \{1:N\}^{\{1:M\}}$ to be computed as a RDD of list, that is `J: RDD[List[Int]]`. For this purpose, method `flatMap` is applied to an iterator describing j,
- From selection map j and the definition of bba $\widetilde{m}_{1:N}$, map to the collection of tuples:

  $$\left(j, \left(Y_{j(i)}, \widetilde{m}_{j(i)} \right)_{i=1:M} \right) .$$

  Only cases, with non zero values for $\widetilde{m}_{j(i)}(Y_{j(i)})$, are considered. Methods `join` and `flatMap` are thoroughly used in this process, resulting in `M: RDD[(List[Int], List[(U,U=>Double)])]`, where generic type `U` is used for encoding subsets,
- The referee function is applied through method `flatMap`, and yields the collection of tuples:

  $$\left((j,X), F\left(X \middle| Y_{j(1:M)}, \widetilde{m}_{j(1:M)} \right) \prod_{i=1:M} \widetilde{m}_{j(i)}\left(Y_{j(i)} \right) \right) ,$$

  as the RDD, `FM: RDD[((List[Int], U),Double)]`,

Reducing Step: In this step, all inner computations are summed up according to summation, $\sum_{Y_{1:M}\subseteq\Omega}$.

- At last RDD `FM` is reduced by key (j,X) with the addition operator. As a result, the combined bba are obtained as the collection of tuples:

  $$\left(j, \left(X, \oplus \left[m_{1:M} | F \right] (X) \right) \right) .$$

  Method `reduceByKey` is used with `+`, yielding `FusedM: RDD[(List[Int], (U,Double))]`.

At this time, the caching strategy is not monitored, and only RDD `J`, defining the selection, and RDD `FM`, defining the final result, are persistent.

3.2.1 Computational Issues

While this approach makes possible a fully distributed computation of the inner elements of the combination during the mapping steps, the amount of cases kept in memory increases exponentially with the number of sources to be combined. Even with a triple combination, the approach consumes a lot of memory, especially when the set of selected combinations is densely connected. In order to address this issue, we propose in next section to bring out and to implement a Markovian property of the rule definition.

3.3 Recursive Formulation

As an instrument for a recursive formulation of the rules, the sets Ψ and Λ are defined for intermediate computations. The rule is then defined on the basis of three finite conditional functions:

$$\begin{aligned}&(\lambda_i, Y_i, m_i) \mapsto \sigma(\lambda_i|Y_i;m_i)\ ,\\ &(\psi_n, \lambda_{1:n}) \mapsto R(\psi_n|\lambda_{1:n})\ ,\\ &(X, \psi_M) \mapsto \pi(X|\psi_M)\ ,\end{aligned}$$

for $X, Y_i \subseteq \Omega$, $\psi_n \in \Psi$, $\lambda_i \in \Lambda$ and $1 \leq i,n \leq M$. Functions σ and π are respectively forward and backward projectors from the space of proposition 2^Ω to the spaces of computation, Λ and Ψ. Function R is a *referee function within the spaces of computation*. In σ, parameter m_i is the bba related to information source i, but any other contextual knowledge could be considered. Based on triplet $[\sigma, R, \pi]$, rule $\oplus[\sigma, R, \pi]$ is defined as a composition of conditional inferences:

$$\begin{aligned}\oplus\left[m_{1:M}|\,\sigma,R,\pi\right](X) \quad &= \sum_{\psi_M\in\Psi} \pi(X|\psi_M)\\ &\sum_{\substack{Y_{1:M}\subseteq\Omega\\ \lambda_{1:M}\in\Lambda}} R(\psi_M|\lambda_{1:M}) \prod_{i=1}^{M} (m_i(Y_i)\sigma(\lambda_i|Y_i;m_i))\ , \quad (5)\end{aligned}$$

for all $X \subseteq \Omega$. Owing to definition (5), it is noticed that, although Ψ and Λ may be *infinite* sets, the summations are actually finite: the values are zeroed except for a finite number of them. On such definition, the main computation burden comes from the conditional inference $R(\psi_n|\lambda_{1:n})$, while other inferences are more or less easily factorized. In order to reduce the computational burden, a Markov hypothesis is made on R by introducing conditional function ρ:

$$R(\psi_{n+1}|\lambda_{1:n+1}) = \sum_{\psi_n\in\Psi} \rho(\psi_{n+1}|\psi_n,\lambda_{n+1}) R(\psi_n|\lambda_{1:n})\ , \quad (6)$$

for $\psi_{1:M} \in \Psi$, $\lambda_{1:M} \in \Lambda$ and $1 \leq n < M$. Under this hypothesis, $\oplus\left[m_{1:M}|\,\sigma,R,\pi\right]$ is computed recursively:

1. For $\lambda_{1:M} \in \Lambda$ and $i = 1:M$, compute projection:
$$\mu_i(\lambda_i) = \sum_{Y_i\subseteq\Omega} m_i(Y_i)\sigma(\lambda_i|Y_i;m_i)\ , \quad (7)$$
2. Compute $\oplus\left[m_{1:M}|R\right]$ recursively within space Ψ:
$$\oplus\left[m_1|R\right](\psi_1) = \sum_{\lambda_1\in\Lambda} \mu_1(\lambda_1) R(\psi_1|\lambda_1)\ , \quad (8)$$
$$\begin{aligned}\oplus\left[m_{1:n+1}|R\right](\psi_{n+1}) &= \sum_{\lambda_{n+1}\in\Lambda} \mu_{n+1}(\lambda_{n+1})\\ &\sum_{\psi_n\in\Psi} \rho(\psi_{n+1}|\psi_n,\lambda_{n+1})\ \oplus\left[m_{1:n}|R\right](\psi_n)\ , \quad (9)\end{aligned}$$
for $\psi_{1:M} \in \Psi$, $\lambda_{1:M} \in \Lambda$ and $1 \leq n < M$,
3. Compute backward projection for all $X \subseteq \Omega$:
$$\begin{aligned}\oplus\left[m_{1:M}|\,\sigma,R,\pi\right](X) \quad &=\\ &\sum_{\psi_M\in\Psi} \pi(X|\psi_M)\ \oplus\left[m_{1:M}|\,\sigma,R\right](\psi_M)\ . \quad (10)\end{aligned}$$

Combined with *map-reduce*, the recursion improves the efficiency of the distributed implementation.

3.3.1 Recursive Definition of D&P Rule

Unprojected definition (4) is not directly compatible with a recursive decomposition. In order to take into account the sources consensuses in a Markov decomposition, the computation set is chosen as the set of multisets of underlying set Ω.

$$m_1 \oplus_{DP} \cdots \oplus_{DP} m_M = \oplus\left[m_{1:M}|\,\sigma_{DP},R_{DP},\pi_{DP}\right]\ ,$$

where σ_{DP} is canonical map from sets to multisets, π_{DP} maps backward from multisets to *top* sets, and R_{DP} evaluates the vote by adding on the multisets:

- $\Lambda_{DP} = \Psi_{DP} = \mathbb{N}^\Omega$,
- $\sigma_{DP}(\lambda_i|Y_i;m_i) = I\left[\lambda_i = \left[I\left[\omega \in Y_i\right]\right]_{\omega\in\Omega}\right]$,
- $\pi_{DP}(X|\psi_M) = I\left[X = \arg\max_{\omega\in\Omega} \psi_M\right]$,
- $R_{DP}(\psi_n|\lambda_{1:n}) = I\left[\psi_n = \sum_{i=1}^{n} \lambda_i\right]$.

Markov decomposition of R_{DP} comes easily:

$$\rho_{DP}(\psi_{n+1}|\psi_n,\lambda_{n+1}) = I\left[\psi_{n+1} = \psi_n + \lambda_{n+1}\right]\ .$$

3.4 Recursive Implementation

Recursive map-reduce implementations of the rules are similar to the non-recursive approach described in section 3.2. But in this case, each recursive step is computed by means of a map-reduce sequence :

Projection Steps: Each bba of $(\widetilde{m}_k)_{k=1:N}$ is computed and mapped into its projection $(\widetilde{\mu}_k)_{k=1:N}$. This projection is done by a map step and a reduce step:

- All bba are computed as a collection of tuples:
$$(k,(Y_k,\widetilde{m}_k(Y_k)))\ ,$$
as `Bba: RDD[(Int, (U,Double))]`. Generic type `U` is used for encoding subsets. For this purpose, `flatMap` is applied to an iterator of the bba,
- From `Bba`, the projected weights are then computed as a collection of tuples:
$$(k,(\lambda,\sigma(\lambda|Y_k;\widetilde{m}_k)\widetilde{m}_k(Y_k)))\ .$$
RDD, `MapBba: RDD[(Int, (L,Double))]`, is obtained by applying `flatMap` to `Bba`. Generic type `L` is used for encoding Λ-parameters,
- `MapBba` is reduced by key (k,λ) with the addition operator. As a result, the projected bba, $\widetilde{\mu}_{1:N}$, are obtained as the collection of tuples:
$$(k,(\lambda,\widetilde{\mu}_k(\lambda)))\ .$$
Method `reduceByKey` is used with `+`, yielding `ProjBba: RDD[(Int, (L,Double))]`,

Recursive Steps: First stage (8) is computed:

- For all prefixes, $j(1)$, of a selection map j, `ProjBba` is mapped to the collection of tuples:
$$\left(j(1),\left(\psi,\widetilde{\mu}_{j(1)}(\lambda)R(\psi|\lambda)\right)\right)\ .$$
`FusMapBba: RDD[(List[Int], (P,Double))]`, is obtained by applying `flatMap` to `ProjBba`. Generic type `P` is used for encoding Ψ-parameters,
- `FusMapBba` is reduced by key $(j(1),\psi)$ with the addition operator. As a result, values $\oplus\left[\widetilde{m}_{j(1)}\middle|R\right]$, are obtained as the collection of tuples:
$$\left(j(1),\left(\psi,\oplus\left[\widetilde{m}_{j(1)}\middle|R\right](\psi)\right)\right)\ .$$
Method `reduceByKey` is used with `+`, yielding `FusProjBba: RDD[(List[Int], (P,Double))]`,
- Set $n\leftarrow 1$,

and subsequent stages (9) are computed until $n=M$:

- Set $n\leftarrow n+1$,
- For all prefixes, $j(1:n)$, of a selection map j, `FusProjBba` is mapped to the collection of tuples:
$$\left(j(1:n),\left(\psi',\widetilde{\mu}_{j(n)}(\lambda)\rho(\psi'|\psi,\lambda)\ \oplus\left[\widetilde{m}_{j(1:n-1)}\middle|R\right](\psi)\right)\right)\ .$$
`FusMapBba: RDD[(List[Int], (P,Double))]`, is obtained by applying `flatMap` to `ProjBba`.
- `FusMapBba` is reduced by key $(j(1:n),\psi)$ with the addition operator. Values $\oplus\left[\widetilde{m}_{j(1:n)}\middle|R\right]$, are obtained as the collection of tuples:
$$\left(j(1:n),\left(\psi,\oplus\left[\widetilde{m}_{j(1:n)}\middle|R\right](\psi)\right)\right)\ .$$
Method `reduceByKey` is used with `+`, yielding `FusProjBba: RDD[(List[Int], (P,Double))]`,

Backward Projection Steps: At last, the combined bba are obtained from `FusProjBba`:

- For all selection maps j, `FusProjBba` is mapped to the collection of tuples:
$$\left(j,\left(X,\pi(X|\psi)\ \oplus\left[\widetilde{m}_{j(1:M)}\middle|\sigma,R\right](\psi)\right)\right)\ .$$
Then, `FM: RDD[(List[Int], (U,Double))]`, is obtained by applying `flatMap` to `FusProjBba`,
- `FM` is reduced by key (j,X) with the addition operator. Combined bba are obtained as the collection of tuples:
$$(j,(X,\oplus[m_{1:M}|\sigma,R,\pi](X)))\ .$$
Method `reduceByKey` is used with `+`, yielding `FusedM: RDD[(List[Int], (U,Double))]`.

4 SOME TESTING CASES

In this preliminary work, some limited tests are made only for the rule of Dubois & Prade. The tests have been done with limited computation power, that is a 6-thread virtual machine with 23 Gio of memory, operated on a i7-4770 GPU with 32 Gio of memory. These tests are a first glimpse of the improvements by our approach. But many optimization works are still in progress and extensions to small clusters of computers are currently investigated.

A collection of bba is first generated randomly on set $\Omega=\{a,b,c,d\}$. Then, any triplet combination of these bba are intended for the computation of Dubois & Prade rule. The following table evaluates the computation time (in seconds) for the entire triplet set with different sizes, $NN=N(N-1)(N-2)/6$, and different computation approaches: 1 thread and non-recursive (1-nr); n threads and non-recursive (n-nr); n threads and recursive (n-r).

NN	560	4960	41664	341376
1-nr	1.7	12	104	–
6-nr	1.23	5.25	37.9	3060
6-r	2.96	7.02	37.3	545

These results confirm the efficiency of the recursive approach for *large* concomitant combination sequences. The recursive approach is however a burden for small sequences. On this preliminary work, the code has not been optimized. For this reason, the table is not significant at this time in comparison with other existing optimized libraries.

Moreover, these tests only considered the performance of simultaneous computation of large set of combinations, and especially, a full set of triple combinations. This implies important intermediate results caching. This case of use is then favourable to

our third, recursive, algorithm, since this approach reduces the caching by definition. But many other aspects of this implementation have to be investigated, in term of performance. Typically, the structure of the set of combinations should be considered for optimizing the strategies of the computation flow. Moreover, an incremental computation of the combinations, may be also investigated through computation flows more complex than map-reduce. From this viewpoint, the reactivity of this parallel computation on possibly complex single combinations is also a piece of performance to be evaluated precisely or optimized in the future, in regards to non-parallel approaches.

5 CONCLUSIONS

In this paper, we proposed a generic distributed processing approach for computing belief combination rules. The approach is based on a map-reduce paradigm, and has been implemented in scala/SPARK. It is derived from the concept of referee function, introduced in a previous work with the aim of separating the definition of the combination rule from its actual implementation. This work has been completed by the proposal of a new recursive formalism for the definition of the rules, and an improved map-reduce generic implementation of the rules processing. Some tests have been made for the rule of Dubois & Prade, which illustrated this computation improvement. More tests will be investigated in the future. Moreover, our intention is to extend this work to general data flow paradigms for computation.

REFERENCES

Dambreville, F. (2009). *Definition of evidence fusion rules based on referee functions*, volume 3. American Research Press.

Dean, J. and Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113.

Dempster, A. P. (1968). A generalization of bayesian inference. *J. Roy. Statist. Soc.*, B(30):205–247.

Dubois, D. and Prade, H. (1986). On the unicity of dempster rule of combination. *International Journal of Intelligent Systems*, 1(2):133–142.

Florea, M., Dezert, J., Valin, P., Smarandache, F., and Jousselme, A. (2006). Adaptative combination rule and proportional conflict redistribution rule for information fusion. In *COGnitive systems with Interactive Sensors*, Paris, France.

Hong, X., Ma, W., Huang, Y., Miller, P., Liu, W., and Zhou, H. (2014). Evidence reasoning for event inference in smart transport video surveillance for video surveillance. In *8th ACM/IEEE INternational Conference on Distributed Smart Cameras*, Prague, Czech Republic.

Lefevre, E., Colot, O., and Vannoorenberghe, P. (2002). Belief functions combination and conflict management. *Information Fusion Journal*, 3(2):149–162.

Liu, W., Miller, P., Ma, J., and Yan, W. (2009). Challenges of distributed intelligent surveillance system with heterogenous information. In *Workshop on Quantitative Risk Analysis for Security Applications*, Pasadena, California.

Martin, A. and Osswald, C. (2007). Toward a combination rule to deal with partial conflict and specificity in belief functions theory. In *International Conference on Information Fusion*, Qébec, Canada.

Shafer, G. (1976). *A mathematical theory of evidence*. Princeton University Press.

Smarandache, F. and Dezert, J. (2005). Information fusion based on new proportional conflict redistribution rules. In *International Conference on Information Fusion*, Philadelphia, USA.

Smets, P. (1990). The combination of evidences in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):447–458.

Zaharia, M., Chowdhury, M., Franklin, M., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proceedings of 2nd USENIX Conference on Hot Topics in Cloud Computing*, Berkeley, CA USA.

Zhou, K., Martin, A., and Pan, Q. (2015a). A similarity-based community detection method with multiple prototype representation. *Physica A: Statistical Mechanics and its Applications*, 438:519–531.

Zhou, K., Martin, A., Pan, Q., and Liu, Z. (2015b). Median evidential c-means algorithm and its application to community detection. *Knowledge-Based Systems*, 74:69–88.

APPENDIX

A Dubois & Prade Rule

Dubois and Prade refined the conjunctive rule by redistributing disjunctively the conflict:

$$m_1 \oplus_{DP} m_2(X) = \sum_{Y_1,Y_2:\begin{cases} Y_1\cap Y_2\neq\emptyset \\ Y_1\cap Y_2=X \end{cases}} m_1(Y_1)m_2(Y_2) + \sum_{Y_1,Y_2:\begin{cases} Y_1\cap Y_2=\emptyset \\ Y_1\cup Y_2=X \end{cases}} m_1(Y_1)m_2(Y_2)\,.$$