PASCAL DJIKNAVORIAN

Fusion d'informations dans un cadre de raisonnement de Dezert-Smarandache appliquée sur des rapports de capteurs ESM sous le STANAG 1241

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise génie électrique
pour l'obtention du grade de Maître ès Sciences (M.Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2008

Résumé

Une analyse comparative de diverses règles de combinaison dans un contexte simple est présentée dans ce mémoire. Nous y retrouverons une identification de l'impact de la fusion de l'information par diverses règles de combinaisons appliquées sur des rapports d'un senseur passif Electronic Support Measures (ESM). Nos objectifs principaux seront donc : d'utiliser la théorie de Dezert-Smarandache (DSmT) pour les ESM, et d'implanter la DSmT pour les ESM. Il faut donc identifier l'impact de la fusion de l'information sous le STANAG 4162 faisant référence au STANAG 1241 lequel énonce cinq catégories d'allégeance pour l'identification. Notre environnement de travail, la théorie de Dezert-Smarandache du possible, paradoxal et du raisonnement neutrosophique est assez récente et a été très peu implémentée ou utilisée avant les travaux couverts par ce mémoire. Elle a été implémentée ici pour un contexte d'environnement ESM.

Abstract

A behavioral comparative analysis of various combining rules in a simple context is presented in this master's thesis. We can find the impact of identification through the information fusion with various combining rules applied on Electronic Support Measures (ESM) output within this thesis. Our main objectives will be : to use the Dezert-Smarandache Theory (DSmT) for ESM, and implement the DSmT for use on ESM reports. Then we will identify the impact of information fusion under the STANAG 4162 which uses the STANAG 1241 to derive an allegiance. Our working environment, the Dezert-Smarandache theory of plausible, paradoxical, and neutrosophic reasoning is quite new and hasn't been used much nor implemented before the work covered by this thesis. It has been implemented here for an ESM environment.

Avant-propos

La fusion d'informations est principalement utilisée en liaison avec des capteurs dans le domaine militaire. Dans ce contexte, la fusion d'informations est donc un outil de décision concernant les objets détectés par des capteurs. D'un aspect purement mathématique, il est possible de voir la fusion d'informations en tant qu'outil mathématique dans le cadre de tout traitement d'informations modélisé pour la prise de décisions. Il serait donc possible de voir cet outil utilisé dans le traitement d'images, que ce soit dans le monde médical, spatial, marin ou autre. Il pourrait être également utilisé dans les modules de décision de systèmes d'intelligence artificielle, au sein de systèmes autonomes, ou dans les systèmes de traitement du langage, par exemple.

Dans un environnement requérant l'utilisation de plusieurs capteurs d'objets se mouvant à différentes vitesses dans différentes directions, la fusion de données peut être utilisée à divers niveaux. On peut la voir utilisée dans l'identification d'objet, dans l'association d'un objet avec une trajectoire, dans un cadre temporel fixe ou dynamique, par le biais de l'utilisation d'un seul ou de multiples capteurs, etc. Dans le cas nous concernant plus spécifiquement, la fusion de données sert à l'identification de l'affiliation d'un objet détecté dans un cadre temporel dynamique utilisant une seule nouvelle information par unité temporelle. Ce type de cas pourrait être vu comme étant un problème de mise à jour de croyance, mais c'est le propre des systèmes de suivis ou d'identification d'une cible solitaire.

Tel que définie dans [29], la fusion d'information à pour but de donner un meilleur estimé de l'état d'une cible par des processus de combinaison d'informations. C'est dans cette optique et suivant la définition de [29] qu'elle est utilisée dans le cadre de notre recherche. L'analyse comparative de règles de combinaisons est faite dans un contexte de combinaison d'information de rapports de senseurs ESM.

Remerciements

S'aventurer dans des domaines inexplorés de la science moderne peut apporter son lot de surprises difficiles et complexes. Cela en augmente d'autant les défis qu'il faut accomplir pour atteindre ses objectifs. Cela permet également, lorsque l'imagination et la créativité sont au rendez-vous, d'apporter quelques contributions novatrices au domaine afin de pouvoir surmonter les problèmes rencontrés. C'est tout cela, la recherche réalisée, les problèmes rencontrées, ainsi que les contributions que vous aurez l'occasion de découvrir dans ce mémoire.

Cet ouvrage n'aurait toutefois pas pu s'être réalisé sans le soutien, la patience, le support et les conseils, morale, académique et financier de divers personnes et organismes. En premier lieu vient mon directeur de recherche, le professeur Dominic Grenier, qui a bien voulu me faire confiance, et qui a cru en moi. Il a ainsi permis la réalisation du travail ayant mené à ce mémoire me donnant judicieux conseils, recommandations, et encouragements. Le Dr. Pierre Valin, mon co-directeur de recherche qui a su répondre à mes diverses interrogations et qui m'a guidé vers un bon sujet de recherche. Lockheed Martin Canada et le CRNSG d'où provient le financement m'ayant supporté.

J'aimerais également souligner la collaboration des membres du groupe de fusion du LRTS, Prof. Dominic Grenier, Dr. Pierre Valin, Dr. Éloi Bossé, Dr. Anne-Laure Jousselme, Dr. Éric Lefebvre, Mme Miroslava Ivanova, Mme Melita Hadzagic, ainsi que Dr. Mihai C. Florea.

Je tiens aussi à remercier les personnes suivantes, qui ont acceptés de prendre le temps d'évaluer mon mémoire et de me donner de précieux commentaires constructifs ; Dr. Jean Dezert, Dr. Mihai C. Florea, Dr. Pierre Valin, et Prof. Dominic Grenier.

Nous avons enfin la famille et les amis qui constituent un entourage bien trop souvent négligé et oublié ; je vous ai probablement négligé dernièrement, mais ne vous ai pas oublié. Merci à la mamma Nejmeh, le pappa Marwan, le ti-frère Édouard, le grand frère Richard, et les amis : Steven, David-Alexandre et Robert. Cheers !

À la mamma et au pappa
Nejmeh & Marwan

Je serai bien aise que ceux qui me voudront faire des objections ne se hâtent point, et qu'ils tâchent d'entendre tout ce que j'ai écrit, avant que de juger d'une partie : car le tout se tient et la fin sert à prouver le commencement. – Descartes

Table des matières

Liste des tableaux

Table des figures

Glossaire

Affiliation Une association à, une appartenance, un rattachement.

Allégeance En référence à la loyauté d'un individu à un autre.

Associatif Une loi de composition interne T sur un ensemble E est dite associative si pour tous ses éléments x,y et z, on a : $xT\left(yTz\right) = \left(xTy\right)Tz$.

Basic belief assignement (bba) $m \ : 2^{\Theta} \rightarrow [0,1]$, donc la masse donnée à un ensemble $A \subseteq \Theta$ suit $m\left(A\right) \in [0,1]$.

Body of evidence (BOE) (voir corps d'évidence)

Cadre de discernement (Θ) Ensemble $\Theta = \{\theta_1, \theta_2, \ldots \theta_n\}$ incluant tout les singletons possibles d'un contexte précis. Les hypothèses de l'ensemble sont supposées exclusives. Aussi, l'ensemble est considéré comme étant exhaustif.

Cadre de raisonnement Ensemble de tous les ensembles possibles composés à partir d'un cadre de discernement suivant les règles de composition d'ensemble dudit cadre de raisonnement. Les compositions d'ensembles possibles sont définis pour chaque cadre de raisonnement.

Cadre de raisonnement de base (voir section 2.3.3)

Cadre de raisonnement DSm Cadre de raisonnement de base avec une extension permettant également des ensembles avec intersections.

Commutatif Une loi de composition interne T sur un ensemble E est dite commutative si pour tous ses éléments x et y, on a : $xTy = yTx$.

Certitude totale L'état de certitude totale survient lorsque l'on est dans une situation où la totalité de la masse attribuable possible, 1, se retrouve localisée sur un seul élément focal autre que l'ignorance totale ou un ensemble d'ignorance partielle. C'est donc dire que $m(A) = 1$, $\forall A \neq I_t$, par exemple.

CIBLE On considère qu'une cible est un avion ou un bateau détecté par un système ESM.

Conflit Un conflit a lieu lorsque l'on obtient une masse issue d'une combinaison sur un ensemble étant une contrainte. Nous avons également un conflit dans le cas où l'on tente de combiner l'avis de sources qui s'opposent.

Contrainte Un ensemble que l'on considère impossible à obtenir.

Contre-mesure Une contre-mesure est une action prise pour s'opposer à une autre action, un effet, un évènement, ou pour les prévenir. La présence d'une contre-mesure peut potentiellement créer un conflit.

Corps d'évidence C'est une fonction de masse où l'on n'a que les éléments focaux et leurs masses correspondantes.

Capteur Un capteur est un appareil de mesure d'une information précise sur un sujet particulier. Il en existe plusieurs types. Il y a par exemple des capteurs mécaniques, optiques, électriques, etc. Chacun des types de capteurs peut avoir différents types de mesures possibles, c'est à dire qu'ils peuvent mesurer une vitesse, un niveau énergétique, un niveau de puissance, une pression, un niveau d'acidité, un niveau de proximité, etc.

Croyance (**Bel**(A)) Évaluation du niveau minimal de confiance, que peut recevoir un ensemble.

Discorde Se dit d'une information conflictuelle, inconsistante.

Electronic Support Measures (ESM) Le ESM est un capteur permettant l'identification, avec un certain niveau d'incertitude, de l'identité de radar(s) émetteur(s) situé(s) sur une cible détectée et qui utilise cette information dans l'optique d'identification de la cible ainsi que son appartenance ami, ennemi ou neutre. Le niveau d'incertitude sur cette identification est très faible normalement. Par contre, le ESM fait cette identification dans un ensemble de possibilités assez large, d'où une possibilité de mauvaise association. De plus, la présence de multiples cibles dans le cône d'identification du détecteur accroît l'incertitude.

Élément focal est l'objet de masse non nul de la fonction de masse. C'est un sous-ensemble A de Θ tel que $m(A) > 0$.

Ensemble Un ensemble est décrit comme étant un regroupement d'objets. Au maximum, il en contient autant que le cadre de discernement. Dans ce dernier cas, cela équivaut également à l'ignorance totale.

Ensemble parent Un ensemble parent B par rapport à un sous-ensemble C est un ensemble tel que $C \subseteq B$.

Ensemble vide ($\emptyset$) Ensemble décrit comme étant le regroupement d'aucun objet.

Ensemble de puissance $\left(2^{\Theta}\right)$ Le cadre de raisonnement issu du cadre de discernement Θ qui permet l'union et inclut l'ensemble vide ($\emptyset$). Avec le cadre de discernement tel que défini ci-haut, on obtient l'ensemble de puissance $2^{\Theta} = \{\emptyset, \{\theta_1\}, \{\theta_2\}, \ldots \{\theta_n\}, \{\theta_1 \cup \theta_2\}, \ldots \{\theta_1 \cup \theta_2 \cup \ldots \cup \theta_n\}, \ldots, \Theta\}$. Il correspond au cadre de raisonnement de base.

Ensemble d'hyperpuissance $\left(D^{\Theta}\right)$ Le cadre de raisonnement issu du cadre de discernement Θ qui permet l'intersection, l'union, et qui inclut l'ensemble vide ($\emptyset$).

Avec le cadre de discernement : $\Theta = \{\theta_1, \theta_2\}$, on obtient l'ensemble d'hyper-puissance $D^\Theta = \{\emptyset\ , \{\theta_1\}\ , \{\theta_2\}\ ,\ \{\theta_1 \cap \theta_2\}\ ,\ \{\theta_1 \cup \theta_2\}\}$. Il correspond au cadre de raisonnement DSm.

Fonction de masse Le regroupement de toutes les paires, masses et objets, appartenant au cadre de raisonnement donné.

Identification Friend-or-Foe (IFF) Un système d'identification qui n'a la possibilité que d'identifier les amis (alliés). Un défaut de réponse ou une mauvaise réponse occasionne soit une réponse variant selon sa configuration parmi ennemi, ami en difficulté, ou neutre en difficulté, par exemple.

IDPC Le IDPC, ou processus de combinaison de données d'identification est une procédure de combinaison de sources différentes pour arriver à une identification unique sur l'identité d'une cible et de son parcours dans le temps.

Ignorance partielle L'ignorance partielle, par rapport à l'ignorance totale, se trouve à être représentée par une union de quelques uns (et non la totalité) des objets membres du cadre de discernement à l'étude ; les objets considérés étant ceux qui interviennent dans la combinaison.

Ignorance totale (I_t) L'ignorance totale d'un cadre de discernement, notée parfois Θ, représente l'objet consistant en l'union de tous les singletons membres du cadre de discernement. $\forall X \in \Theta,\ \cup_T = \bigcup_{i=1}^{|\Theta|} X_i = I_t$

Imprécision Une imprécision peut être vue comme étant une erreur dans une mesure.

Incertitude Une incertitude peut être vue comme étant une croyance et/ou un doute et/ou confusion à propos d'une mesure.

Intersection L'intersection est un opérateur mathématique à deux arguments et une sortie. L'intersection des ensembles A et B est l'ensemble de tous les éléments qui appartiennent à la fois à A et B uniquement. Elle est désignée par $A \cap B$, où $A \cap B = \{x |\, (x \in A) \wedge (x \in B)\}$.

Intersection totale Pour un cadre de discernement donné, l'intersection totale est l'objet qui correspond à l'intersection de tous les singletons dudit cadre. $\forall X \in \Theta,\ \cap_T = \bigcap_{i=1}^{|\Theta|} X_i$

Intervenant On considère les intervenants d'une opération de combinaison comme étant les objets pour lesquels on applique une combinaison. On considère les intervenants d'un conflit comme étant les deux objets (ou plus) lors de l'opération de combinaison résultant en une masse conflictuelle.

Masse Le poids associé au niveau de certitude attribué à un objet. La masse d'un objet est à la théorie de l'évidence ce que la probabilité d'une hypothèse est à la théorie des probabilités. La masse associée à un objet se décrit mathématiquement par $m : 2^\Theta \rightarrow [0, 1]$. Pour l'objet $\emptyset$, la valeur de masse est de 0 ; et la somme des masses pour tous les objets d'un cadre de discernement donnée est unitaire. (Voir aussi *Basic belief assignement.*)

Neutrosophie Une nouvelle branche de la philosophie considérant non seulement le vrai et le faux mais également des éléments neutres ainsi que des intermédiaires entre ces trois états. Une logique neutrosophique, ainsi que des statistiques et probabilités neutrosophiques découlent de ce principe.

Noyau de Θ ($\mathcal{K}$) L'ensemble de tous les éléments focaux de Θ.

Objet Un objet est une composition de singletons via les opérateurs d'intersection ou d'union à partir de singletons identiques ou différents. Un singleton est donc également un objet. Les objets sont membres d'un ensemble qui peut en contenir aucun. Notez qu'un objet peut également être l'ensemble vide selon les règles de composition du cadre de raisonnement.

Objet parent Un objet parent B est défini tel qu'il respecte $B \supseteq A$, où A est le sous-objet de B.

OTAN Acronyme de « Organisation du Traité de l'Atlantique Nord ». Organisme ayant vu le jour le 4 avril 1949 avec la signature d'un traité. L'OTAN a établi un système de défense collectif où les pays membres acceptent de s'entraider au niveau de la défense lors d'une attaque extérieure.

Plausibilité ($\mathbf{Pl}(A)$) Évaluation du niveau maximal de certitude, de confiance, que peut recevoir un ensemble.

Singleton $\{\theta\}$ Un singleton est défini comme suit : $\forall\theta, \exists S/\forall x, (x \in S) \Leftrightarrow (x = \theta)$ où S est un ensemble, et θ, l'unique objet qu'il contient. (Un objet de cardinalité de 1.)

Source d'évidence Source produisant après un pré-traitement, un corps d'évidence.

Sous-objet Un sous-objet A est défini tel qu'il respecte $A \in B$, où B est l'objet parent de A.

Sous-ensemble Un sous-ensemble A est défini tel que si tous les objets composant A sont également membres de B, alors A est un sous-ensemble de B. On désigne la relation entre A et B comme suit $A \subseteq B$.

STANAG Abréviation donnée par l'OTAN pour Standardization Agreement.

Cet accord met en place des processus, procédures, termes, et conditions pour l'équipement ou les procédures techniques militaires mis en commun parmi les membres de l'alliance Nord Atlantique.

Union L'union est un opérateur mathématique à deux arguments et une sortie. L'union des ensembles A et B est l'ensemble de tous les éléments qui appartiennent soit à A ou B ou les deux. Elle est désignée par $A \cup B$ où $A \cup B = \{x | (x \in A) \vee (x \in B)\}$.

Union totale Pour un cadre de discernement donné, l'union totale est l'objet qui correspond à l'union de tous les objets du cadre de discernement. (Voir aussi *Ignorance totale.*)

Liste des symboles

$m(\cdot)$	Masse d'un ensemble.
Θ	Cadre de discernement.
2^Θ	Cadre de raisonnement de l'ensemble de puissance.
D^Θ	Cadre de raisonnement de l'ensemble d'hyperpuissance.
G^Θ	Cadre de raisonnement généralisé.
$\emptyset$	Ensemble vide, ensemble ne comportant aucun élément.
I_t	Ensemble d'ignorance totale.
$\mathcal{K}$	Noyau de l'ensemble de discernement.
$\mathcal{M}^0$	Modèle de Shafer.
$\mathcal{M}^f$	Modèle libre.
$\mathcal{M}$	Modèle hybride.
m_Y	Masse résultant d'une combinaison de Yager.
m_{DP}	Masse résultant d'une combinaison de Dubois et Prade.
m_S	Masse résultant d'une combinaison de Smets.
m_{Ina}	Masse résultant d'une combinaison de Inagaki.
$m_\vee$	Masse résultant d'une combinaison disjonctive.
$m_\wedge$	Masse résultant d'une combinaison conjonctive.
m_{DS}	Masse résultant d'une combinaison de Dempster.
$m_k(\cdot)$	Masse de l'objet $\cdot$, selon la k-ième source.
$Bel(\cdot)$	Fonction de croyance.
$Pl(\cdot)$	Fonction de plausibilité.
ESM	Electronic Support Measures.
IFF	Identification Friend-or-Foe.
OTAN	Organisation Trans-Atlantique Nord, ou NATO en anglais.
STANAG	Accord de Standardisation de l'OTAN.

Chapitre 1

Introduction

Un problème sans solution est un problème mal posé. – Albert Einstein

1.1 Mise en contexte

N'étant pas parfaitement fiables et n'analysant qu'un aspect ou une caractéristique d'une cible détectée, les capteurs requièrent que l'on tienne compte de plus d'une source d'informations pour assurer un certain niveau de certitude quant à l'identification et l'appartenance de la cible détectée. Ainsi, les responsables de prises de décisions ont sous la main des relevés d'informations provenant de diverses sources au sujet de diverses cibles et groupes de cibles détectés. Nonobstant la surcharge potentielle d'informations lorsque de multiples informateurs donnent leurs opinions quant à l'identité d'une cible détectée, il y a le fait que ces opinions sont bien souvent incertaines ou imprécises ou incomplètes sinon erronées. Il est nécessaire que le processus de prise de décisions considère le niveau de fiabilité de ces diverses sources pour atteindre un seuil de qualité. Un des moyens pour atteindre cet objectif est la combinaison de l'information dans un cadre de raisonnement pour construire des propositions qui s'orientent dans la direction de la vérité. Là se trouve le but de la fusion d'information : réunir plusieurs informations de qualités diverses pour en obtenir une version simplifiée, unifiée, de meilleure qualité. Il faut toutefois faire bien attention par rapport à la fusion d'information. Dans [17] on dit que la fusion ne remplacera pas un capteur idéal et donc, on ne pourrait jamais être meilleur avec une fusion de sources multiples de qualité douteuse, comparativement à une source unique de qualité exceptionnelle.

Les diverses règles de combinaison travaillent différemment sous différentes conditions d'utilisation. Le cas précis auquel nous désirons nous attaquer consiste à étudier et à analyser le comportement et les performances de quelques règles dans le contexte d'une cible détectée par des Electronic Support Measures (ESM) en utilisant le cadre particulier de raisonnement DSm. Nous avons choisi ce cadre de raisonnement pour plusieurs raisons :

1. un rapport ESM peut seulement déclarer qu'une cible est 'amie', 'neutre' ou 'ennemie', mais le preneur de décisions voudrait aussi des situations plus précises telles que 'suspecte' ou 'présumée amie' qui sont le résultat d'intersection entre 'neutre' et 'ennemie' ou 'neutre' et 'amie' respectivement[1]. Or, le raisonnement DSm permet d'introduire les intersections d'objets.
2. Le cadre de raisonnement DSm a très peu été utilisé à ce jour. Il serait donc intéressant de comprendre et d'analyser son comportement en le comparant avec un autre cadre de raisonnement plus classique en fusion de données, c.-à-d. : la théorie de l'évidence.

[1] 'Suspecte' est également possible par le résultat de l'intersection de 'amie' et 'ennemie'.

1.2 Objectifs visés

Les travaux entrepris ont pour tâche de réaliser une combinaison d'informations sur l'allégeance provenant de capteurs ESM dans le cadre de raisonnement relativement nouveau DSm, selon la théorie de Dezert-Smarandache (DSmT). Une analyse permettrait de mieux comprendre le comportement de diverses règles dans le cadre de raisonnement DSm. Comme la règle de Dempster-Shafer (DS) dans la théorie de l'évidence (Dempster-Shafer Theory (DST)) demeure un incontournable malgré certains désavantages déjà connus, on comparera les résultats avec ceux fournis par cette règle de combinaison. Il faut savoir que la règle de DS continue d'être bien appréciée par les adeptes des théories non bayésiennes à cause de sa qualité d'associativité même s'il arrive qu'il y ait difficulté lors de combinaison d'informations à niveau de conflit élevé[2]. On s'interroge toutefois quant à savoir si les nouvelles règles dans un cadre de raisonnement DSm sont en mesure de réaliser les tâches demandées de façon plus efficace tout en évitant l'inconvénient de la combinaison de DS [38, 39, 40]. Notre recherche consiste donc de réaliser la combinaison d'informations sur l'allégeance provenant de capteurs ESM dans le cadre de raisonnement DSm et de comparer les résultats en utilisant des règles de combinaison récentes dans DSm avec deux règles classiques, la Dempster-Shafer et la méthode du vote majoritaire. Nous en déterminerons les meilleures et moins bonnes, avec leurs bons et moins bons côtés. En d'autres termes, nous aurons à comparer, pour le contexte choisi, les règles de combinaison sous DSmT par rapport aux règles de combinaison sous la DST.

1.2.1 Objectifs principaux

Nos objectifs principaux sont donc les suivants :

1. Utiliser la DSmT pour les ESM en utilisant les définitions d'allégeance du STANAG 1241.
2. Implanter la DSmT pour un cadre de raisonnement ayant trois objets.

[2]Notez toutefois que [20, 15] démontrent que ce problème avec la DS n'est dû qu'à un problème au niveau de la modélisation de l'information à combiner.

1.2.2 Objectifs secondaires

Nos objectifs secondaires deviennent ainsi les suivants :

1. Développer le code nécessaire pour l'utilisation de la DSmT ;
2. Valider la théorie et le code pour les ESM sous la DSmT ;
3. Adapter la théorie et le code nécessaire sous la DSmT ;
4. Comparer la règle de combinaison Dempster sous la DST et DSmH sous la DSmT pour des ESM ;
5. Tenter des optimisations au niveau de la performance opérationnelle.

1.3 Structure du mémoire

Après ce chapitre d'introduction, nous entreprendrons dans le chapitre 2 un tour de l'environnement mathématique englobant le coeur du sujet. On y verra brièvement la base statistique et la théorie de l'évidence d'où proviennent les principes derrière les théories et cadres de raisonnement explorés dont le cadre de raisonnement DSm. Ce sera suivis d'une exploration des principes et des fondements de la fusion d'informations juste avant de voir les règles de combinaison que l'on désire explorer dans le cadre de ce travail. En fin de chapitre, on verra les détails du problème à résoudre.

Le chapitre 3 porte sur les recherches et développements ayant été nécessaires pour la résolution de la problématique présentée, notamment, l'application du STANAG 1241 sous la DSmT. On procèdera en premier lieu avec la présentation de la méthodologie de comparaison des règles de combinaison que l'on a utilisée. On poursuivra avec l'établissement de critères de performance. Nous verrons ensuite la conception, l'implantation et le fonctionnement du système de simulations.

Le chapitre 4 porte sur les problèmes rencontrés ainsi que sur les solutions adoptées pour les résoudre. C'est au sein du chapitre 5 que l'on passe en revue les résultats des simulations et qu'on en fait une analyse avant d'en tirer des conclusions. On aura alors accompli le tour de la problématique, de la solution adoptée et des résultats obtenus. Les conclusions de la recherche qui seront tirées, seront présentées au chapitre 6 après un court rappel des objectifs.

Chapitre 2

Revue de la littérature

Douter de tout ou tout croire, ce sont deux solutions également commodes qui, l'une et l'autre, nous dispensent de réfléchir. – Henri Poincaré

2.1 Des statistiques à la théorie de l'évidence

Dans la grande famille des théories utilisées pour le traitement de l'information incertaine, il existe les anciennes théories, bien rodées, applicables à divers contextes. Par contre, ces théories ne sont pas en mesure de bien traiter les informations vagues, incertaines, conflictuelles dans tous les cas. D'où la raison d'être des plus récentes théories, encore en développement, qui peuvent s'appliquer à des contextes particuliers difficilement représentables sous les plus anciennes théories, telle la théorie des probabilités de Bayes, par exemple.

Dans le cas présent, on se concentre surtout sur des modèles permettant le raisonnement et la décision sous incertitude : imprécision, confusion, ambiguïté, ou encore non-spécificité, flou et discorde. Un des modèles mathématiques permettant de travailler sous incertitude s'appelle la théorie de l'évidence. Elle admet la discorde comme la théorie des probabilités, mais gère aussi la non-spécificité comme la théorie des ensembles, ce qui permet de réaliser une étude mieux adaptée aux situations réelles. C'est le cas, par exemple, lors d'un manque d'informations ou lorsque les sources sont mises en doute. Intégrant l'incertitude menant à la discorde, la théorie de l'évidence diffère de la théorie bayésienne des probabilités, entre autres, par le fait que les probabilités mesurent l'incertitude par entité individuelle précise par le biais d'une fonction de probabilité.

Il existe en fait, divers modèles pour la représentation de la connaissance imparfaite[1]. Voici une liste où l'on retrouve les différentes imperfections que peut subir une information. Il s'agit de l'exemple d'un joueur de baseball, nommé Roberto, qui a exactement 0.359 de moyenne au bâton au cours d'une saison. On demande cette information à différentes personnes et l'on obtient :

- Information parfaite : Roberto a obtenu 0.359.
- Information imprécise : Roberto a obtenu entre 0.100 et 0.400.
- Information confuse : Roberto a obtenu 0.359, mais je n'en suis pas sûr.
- Information vague ou floue : Roberto a obtenu une moyenne tournant autour de 0.360.
- Information probabiliste ou discordante : Roberto a 50% de chances d'avoir 0.359.
- Information non-spécifique : Roberto ou Miguel a obtenu 0.359.
- Ignorance partielle : Roberto avait 0.310 à la mi-saison.
- Ignorance totale : On a absolument aucune idée de la moyenne de Roberto.

[1]Par opposition à la connaissance parfaite, la connaissance de la vérité sur un évènement.

L'information probabiliste et confuse peut être représentée par la théorie des probabilités [2]. Il est également possible de la représenter par la théorie de l'évidence [23]. L'information imprécise, floue, confuse ou non spécifique peut être représentée par la théorie des ensembles flous [10, 38]. Il existe aussi d'autres façons de classifier l'imperfection des informations [27, 1]. Par exemple, la théorie des possibilités regroupe l'aspect flou et l'aspect aléatoire des informations. Il existe aussi des théories appropriées pour la représentation de chacun de ces cas, mais souvent contraintes à un ou deux cas uniquement. Des travaux de recherche ont tenté d'établir un modèle généralisant toutes ces théories, de les unifier, mais ce n'est qu'un début [12].

La typologie de l'incertitude a entre autres été étudiée dans [3]. Cette source, suite à l'étude de nombreux modèles de l'incertitude, propose la typologie que l'on représente dans la figure 2.1. Cette figure montre la relation entre différents types d'imperfections possibles selon le modèle établit dans [3] .

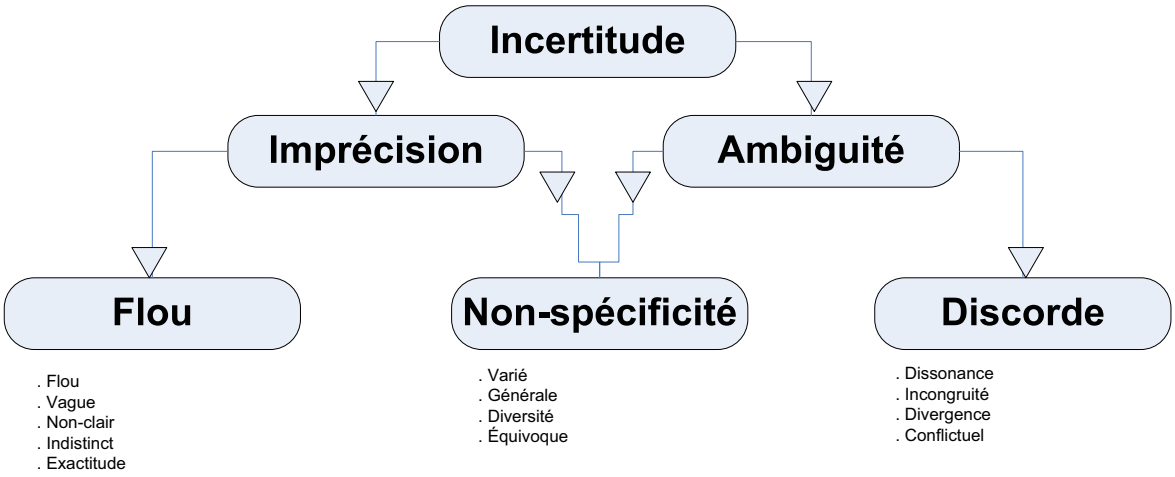


FIG. 2.1 – Typologie de l'incertain, tel que proposée par [3]

Les travaux de Dempster et Shafer [5, 23] sont à la base du développement de la théorie de l'évidence. Ces travaux ont débuté avec Dempster sur les bornes inférieures et supérieures d'une famille de distributions de probabilités. C'est à partir de ce formalisme mathématique que les fonctions de croyance se sont avérées utiles pour la modélisation de connaissances incertaines suite aux travaux de Shafer, qui a suivi un cours donné par Dempster. La démonstration, de manière axiomatique, de l'utilité de la fonction de croyance par rapport à la théorie de la probabilité a été faite par Smets [27].

Dans [1], on définit, entre autres, le vocabulaire suivant :

– **données** : nombres ou symboles non reliés qui représentent des cibles ou entités avec un certain niveau de fiabilité ou de croyance.

- **faits** : données reliées.
- **informations** : faits mis en contexte.
- **connaissance** : un assemblage d'informations qui forme des croyances justifiées, informations mis en contexte.
- **expérience** : principalement de l'interaction avec soi-même et le monde extérieur, incluant le traitement intérieur de la connaissance et des croyances en découlant.

On y apprend également que l'information, une fois traitée et interprétée devient connaissance, qui quant à elle, une fois mise à l'épreuve, évolue, puis peut être remise en traitement et réinterprétée avec l'information courante.

Dans [32] on donne également quelques définitions importantes dont :

- **certitude** : La certitude se caractérise par le fait que l'on est en mesure d'assigner une probabilité unitaire à un évènement A. C'est une situation où l'on parierait sur l'occurrence de l'évènement A, quelque soit la mise du pari.
- **incertitude** : L'incertitude est l'absence de certitude. On ne peut assigner une probabilité unitaire sur la véracité de l'occurrence de l'évènement A en question.
- **incertitude déterminée** : L'incertitude déterminée désigne la situation où l'on a, parmi deux choix incertains, une préférence pour l'un des deux choix, que ce soit par une différence au niveau des probabilités d'occurrences d'évènements, ou simplement par l'impression du décideur que l'un des deux choix est le meilleur.
- **incertitude indéterminée** : L'incertitude indéterminée désigne la situation où l'on n'est pas en mesure d'avoir une préférence pour l'un de deux choix, qui ne sont pas équivalents pour le décideur, mais pour lesquels il ne s'est pas fait une idée sur le bon choix à faire. Il y a présence d'indécision.
- **précision** : La précision réfère aux propriétés mathématiques de linéarité, d'additivité, d'unicité, et d'exhaustivité.
- **imprécision** : L'imprécision fait référence à la propriété mathématique de non-linéarité des prévisions inférieures et supérieures, de la non-additivité des probabilités inférieures et supérieures, la non-exhaustivité d'une classe de paris désirables ou de préférences d'ordre d'arrivée, ainsi que la non-unicité des prévisions linéaires dominantes.

Dans la théorie de Dempster-Shafer, qui correspond à la théorie mathématique de l'évidence, le niveau d'incertitude d'occurrence liée à un évènement est basé sur des fonctions de croyance et du raisonnement plausible afin de combiner l'information pour pouvoir déterminer le niveau d'incertitude d'occurrence liée à un évènement. Des transformations sont possibles afin de passer d'un modèle de représentation de l'incertitude à un autre [12]. La théorie de l'évidence peut être considérée comme étant un cas particulier de la théorie des ensembles aléatoires [12]. La théorie de Dempster-Shafer permet également de considérer la confiance que l'on met dans les probabilités attribuées à chacune des possibilités d'évènement d'un ensemble d'évènements possibles.

2.2 Fusion d'informations

Jusqu'en 1986 la fusion d'informations était utilisée par divers regroupements sans aucune normalisation terminologique, ce qui pouvait évidemment porter à confusion, et nuire à l'avancement potentiel de la recherche. En 1986 un regroupement, le *Joint Directors of Laboratories* (JDL) fut fondé et chargé de la responsabilité de faciliter la compréhension et les échanges efficaces entre les divers intervenants directs ou indirects dans les techniques de fusion d'informations. Une de leurs tâches fut de définir la fusion d'informations. Selon [16], la définition que fut faite par le JDL dans [35] sur la fusion de données, se lit comme suit :

> *A process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats, and their significance. The process is characterized by continuous refinements of its estimates and assessments, and the evaluation of the need for additionnal sources, or modification of the process itself, to achieve improved results.*

Ayant été longtemps considérée comme étant trop restrictive pour diverses raisons, la définition fut changée en 1998, suite à une conférence de l'OTAN tenue à Québec. La nouvelle définition émise dans [29] se lit comme suit :

> *Data fusion is the process of combining data or information to estimate or predict entity states.*

On considèrera la fusion ainsi, comme une méthodologie de combinaison d'informations pour une meilleure estimation de l'état de l'entité observée. Il s'agit de la définition de 1998 de [29], mais il ne faut pas oublier que la définition est en constante évolution.

2.2.1 Type de fusion

Le JDL a également travaillé à l'élaboration d'un modèle structuré de représentation de la fusion d'informations. Diverses discussions et analyses sur le sujet sont entre autres présentées dans [16, 33]. Pour distinguer les divers types de fusion, le JDL a réparti la fusion sous différents niveaux. Chacun des niveaux se distinguant des autres de par ses objectifs, ses méthodologies de fonctionnement, ses outils mathématiques utilisés, le contexte et le type d'informations. La figure 2.2 montre le flot d'informations entre les niveaux ainsi que la structure du modèle du JDL[2]. Bien évidemment, il existe plusieurs variantes du modèle, adaptées selon les besoins.

[2]La figure a été inspirée des figures du chapitre 2 de [17]

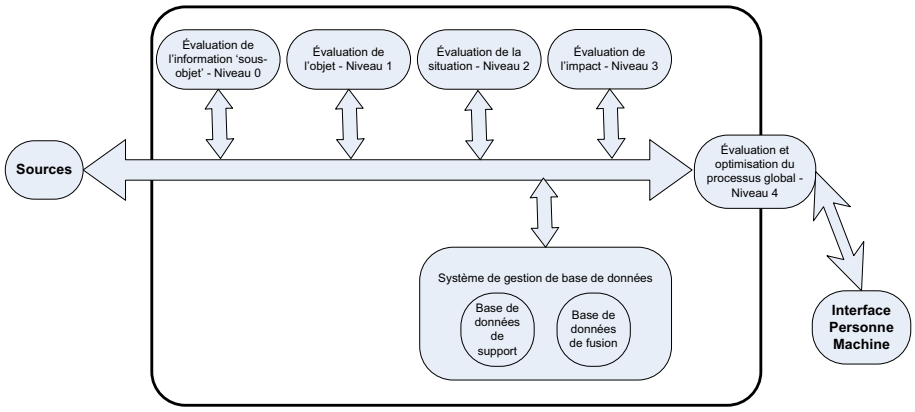


FIG. 2.2 – Modélisation de la fusion d'information et de ses divers niveaux

Brièvement, les niveaux se définissent comme suit :

- **niveau 0 : Évaluation de l'information sous-objet** Détection, estimation et prédiction de l'état de l'objet observé à l'aide d'association, caractérisation et classification des informations au niveau du pixel ou du signal.
- **niveau 1 : Évaluation de l'objet** Détection, estimation et prédiction de l'état d'entité à l'aide d'inférences sur les observations.
- **niveau 2 : Évaluation de la situation** Détection, estimation et prédiction de l'état d'entité à l'aide d'inférences sur les relations entre les entités.
- **niveau 3 : Évaluation de l'impact** Détection, estimation et prédiction des effets (points forts, points faibles) sur la situation par des actions prévues, ou estimées des participants.
- **niveau 4 : Évaluation et optimisation du processus global** Calibration et optimisation de l'acquisition et le traitement des données pour le support des objectifs de mission.

Afin de mieux se situer, il faut savoir que l'objet des travaux de recherche présentés dans le cadre de ce mémoire se situe au niveau 1 de la fusion.

2.2.2 Cadre de raisonnement de base

Le cadre de raisonnement utilisé est celui de la théorie de l'évidence décrit dans [23] par Shafer. Il utilise une version étendue du cadre de discernement Θ qui est un ensemble incluant tous les n objets possibles dans un contexte précis $\Theta = \{\theta_1, \theta_2, \ldots \theta_n\}$. Les hypothèses de l'ensemble sont supposées exclusives. Aussi, l'ensemble est considéré comme étant exhaustif.

Le cadre de raisonnement 2^{Θ} se définit comme étant l'ensemble de tous les sous-ensembles possibles composés à partir des objets d'un cadre de discernement Θ qui permet l'union et inclue l'ensemble vide ($\emptyset$). On désigne le cadre de raisonnement 2^{Θ}comme étant l'ensemble de puissance. Avec le cadre de discernement $\Theta = \{\theta_1, \theta_2, \ldots \theta_n\}$, nous obtenons l'ensemble de puissance :

$$2^{\Theta} = \left\{\emptyset, \{\theta_1\}, \{\theta_2\}, \ldots \{\theta_n\}, \{\theta_1 \cup \theta_2\}, \ldots \{\theta_1 \cup \theta_2 \cup \ldots \cup \theta_n\}, \ldots, \Theta\right\} \tag{2.1}$$

2.2.3 Fonction de masse et ses propriétés

La fonction de masse d'un ensemble peut être vue comme étant le niveau de croyance, de confiance, accordé à cet ensemble en particulier.

Soit un cadre de discernement : $\Theta = \{\theta_1, \theta_2, \ldots \theta_n\}$.

Soit un cadre de raisonnement 2^{Θ} dans lequel on retrouve un ensemble A qui est un élément membre du cadre, donc $A \in 2^{\Theta}$ ou encore $A \subseteq \Theta$.

La fonction de masse $m(\cdot)$ dans la DST, est une fonction $m : 2^{\Theta} \rightarrow [0, 1]$ répondant à deux exigences qui trouvent correspondance dans la théorie des probabilités.

1. Respect de l'hypothèse de monde fermé où la masse de l'ensemble vide doit être nulle :
$$m[\emptyset] = 0 \tag{2.2}$$
2. La somme des masses de tout ensemble possible correspond toujours à l'unité :
$$\sum_{\forall A \subseteq 2^{\Theta}} m(A) = 1 \tag{2.3}$$

La i^{ieme} bba[3] d'une source qui fournit une information d'évidence est notée m_i.

[3]Basic Belief Assignement (bba) se définit comme $m : 2^{\Theta} \rightarrow [0, 1]$, donc la masse donnée à un ensemble $A \subseteq \Theta$ suit $m(A) \in [0, 1]$.

2.2.4 Méthodes de combinaisons

Méthode statique

La méthode statique est une méthode de combinaison où l'on procède en bloc après avoir reçu toutes les informations, ou une quantité suffisante, pour les combiner en une étape. Cette étape augmente clairement en complexité à mesure que la dimension du bloc d'informations s'accroît.

Méthode dynamique

La méthode dynamique est plus simple et permet le travail en temps réel. Elle procède simplement en ne faisant que des combinaisons par paires de corps d'évidence.

2.2.5 Représentation des ensembles dans les diagrammes de Venn

Pour la représentation des ensembles dans les diagrammes de Venn nous reprenons la notation utilisée en [25]. Par exemple, pour le cas de la figure 2.3 qui représente D^{Θ}avec un cardinal de 3 sous un modèle[4] libre $\mathcal{M}^f$, nous avons i qui représente la partie appartenant à θ_i seulement, où ij représente celle qui appartient à θ_i et θ_j seulement[5] et ainsi de suite. La zone grise de la figure 2.3 représente 1, plus spécifiquement la part de θ_1 excluant ses intersections avec d'autres singletons.

2.2.6 Vote majoritaire

Le moyen le plus simple de prendre une décision en tenant compte de plusieurs sources est simplement de procéder à un vote majoritaire. Le vote majoritaire est utilisé en particulier lors d'élections et se présente sous plusieurs versions. La plus simple consiste à sommer les différentes occurrences d'évènements. Celui ayant reçu le plus grand nombre de votes est désigné gagnant.

[4] voir page 30

[5] Ainsi, tel qu'expliqué dans [25] $ij \neq \theta_i \cap \theta_j$, mais plutôt à $\theta_i \cap \theta_j - \theta_i \cap \theta_j \cap \theta_k$ dans un cadre de discernement de cardinalité de 3.

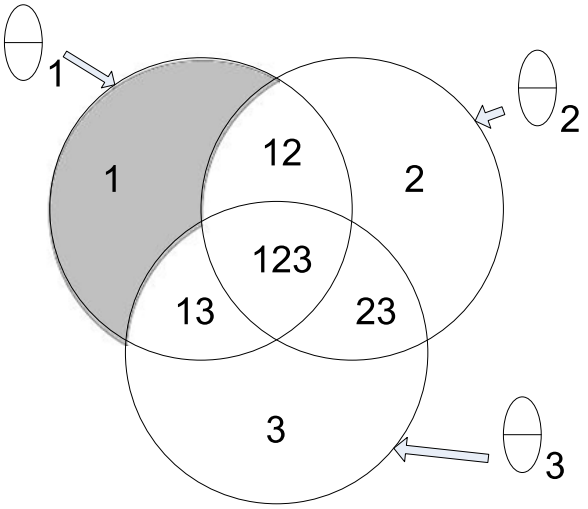


FIG. 2.3 – Diagramme de Venn dans D^{Θ}avec un cardinal de 3

Vote majoritaire à moyenne mobile

Au lieu de simplement prendre le vote en sommant toutes les occurrences d'évènements, cette méthode consiste à moyenner/voter sur les x dernières[6] itérations ou occurrences d'évènements incluant celle de l'itération où l'on procède au vote.

Système d'élection

Il y a principalement deux systèmes d'élection, le plurinominal, et l'uninominal. Le premier peut être utilisé en *multitracking*[7] ainsi qu'en mode *single tracking*[8], alors que le second système ne fonctionne qu'en mode *single tracking*. En mode uninominal, il ne peut y avoir qu'une seule décision à la fois, alors que le mode plurinominal permet plusieurs décisions simultanées. Dans le cas de ce mémoire, seulement le mode *single tracking* est concerné, et les types de vote de cette catégorie seront détaillés.

[6]La fenêtre d'opération peut être de durée différente selon le choix de l'opérateur. Nous utilisons une fenêtre de cinq itérations dans notre cas.

[7]Ce terme signifie qu'il peut y avoir un nombre indéterminé de cibles qui sont suivis par un seul ou plusieurs capteurs de façon simultanée, de façon synchronisée ou pas. L'équivalent en terme d'élection ou de vote, c'est le vote de plusieurs paliers gouvernementaux ou le référendum à plusieurs questions.

[8]Ce terme signifie qu'il n'y a qu'une seule cible à la fois suivie par un seul ou plusieurs capteurs. L'équivalent en terme de vote, c'est le vote uninominal.

Types de vote

Il y a trois principales catégories de types de vote dans le système à élection uninominal : ceux à vote simple, ceux par système de classement, et enfin ceux par système à pondération.[9]

Exemple 2.1 Combinaison d'informations avec le vote majoritaire

Soit un décideur qui est en mesure d'utiliser trois capteurs[10] *d'informations statistiquement indépendantes et de même fiabilité.*

Le capteur 1 *donne l'information suivante :*
$\{m(\theta_1) = 0.60; m(\theta_2) = 0.20; m(I_t) = 0.20\}$.

Le capteur 2 *donne l'information suivante :*
$\{m(\theta_1) = 0.20; m(\theta_2) = 0.20; m(I_t) = 0.60\}$.

Enfin, le capteur 3 *donne quant à lui l'information suivante :*
$\{m(\theta_1) = 0.50; m(\theta_3) = 0.40; m(I_t) = 0.10\}$.

Le vote majoritaire ne considère que les premiers choix de toutes les BOE (corps d'évidence) considérés. Ce cas-ci est plutôt simple : le BOE 1 *vote*[11] *pour* θ_1*, le BOE* 2 *vote pour* I_t*, puis le BOE* 3 *vote également pour* θ_1*. La majorité ayant voté pour* θ_1*, la décision sera* θ_1*.*

[9] Plus d'informations peuvent, entre autres, être retrouvé dans [21].

[10] $m(K)$ représente la masse de l'ensemble K, et I_t, l'ensemble représentant l'ignorance totale, ne pas hésiter à consulter le glossaire et la table des symboles en cas de doute.

[11] La décision est basée sur la masse maximale pour l'instant.

2.3 Règles de combinaisons classiques

2.3.1 Combinaison conjonctive

La combinaison conjonctive est une règle de combinaison élémentaire consistant à prendre les sommes de toutes les possibilités de conjonction entre tous les objets. En fait, la combinaison de deux objets consiste à prendre le produit des masses des deux objets à combiner et à attribuer le résultat du produit à l'objet résultant de l'intersection des deux objets. Une fois cette opération réalisée, on regroupe les éléments communs.

$$q(A) = m_1 \wedge m_2 = \sum_{\substack{B \cap C = A \\ B, C \subseteq \Theta}} m_1(B)\, m_2(C) \qquad \forall A \subseteq \Theta \tag{2.4}$$

Exemple 2.2 Combinaison d'informations avec la règle conjonctive

Reprenant les informations fournies à l'exemple 2.1, on procède à la combinaison conjonctive par la méthode dynamique. Avant de débuter, on place dans une table, les informations à manipuler.

TAB. 2.1 – Évènements de trois corps d'évidence à combiner

	$m_1(A)$	$m_2(A)$	$m_3(A)$
$\{\theta_1\}$	0.6	0.2	0.5
$\{\theta_2\}$	0.2	0.2	0.0
$\{\theta_3\}$	0.0	0.0	0.4
$\{I_t\}$	0.2	0.6	0.1

La première étape consiste à faire le produit orthogonal entre les informations du BOE 1 *avec ceux du BOE* 2*. Selon l'équation (2.4) on obtient la table 2.2.*

Soit $m_{12}(\theta_1) = 0.52$*,* $m_{12}(\theta_2) = 0.20$*,* $m_{12}(\theta_1 \cap \theta_2) = 0.16$ *et* $m_{12}(I_t) = 0.12$*.*

La seconde étape consiste à combiner les résultats de la combinaison de BOE 1 *et BOE* 2 *avec les informations de BOE* 3*. Il en résulte la table 2.3.*

TAB. 2.2 – Résultats d'une étape intermédiaire de la combinaison de BOE 1 et de BOE 2 par le biais de la règle conjonctive

	$m_1(\theta_1)$ 0.6	$m_1(\theta_2)$ 0.2	$m_1(I_t)$ 0.2
$m_2(\theta_1)$ 0.2	θ_1 0.12	$\theta_1 \cap \theta_2 = \emptyset$ 0.04	θ_1 0.04
$m_2(\theta_2)$ 0.2	$\theta_1 \cap \theta_2 = \emptyset$ 0.12	θ_2 0.04	θ_2 0.04
$m_2(I_t)$ 0.6	θ_1 0.36	θ_2 0.12	I_t 0.12

TAB. 2.3 – Résultats de la combinaison de BOE 3 avec les résultats du tableau 2.2 par le biais de la règle conjonctive

	$m_{12}(\theta_1)$ 0.52	$m_{12}(\theta_2)$ 0.20	$m_{12}(\theta_1 \cap \theta_2) = \emptyset$ 0.16	$m_{12}(I_t)$ 0.12
$m_3(\theta_1)$ 0.5	θ_1 0.26	$\theta_1 \cap \theta_2 = \emptyset$ 0.10	$\emptyset$ 0.08	θ_1 0.06
$m_3(\theta_3)$ 0.4	$\theta_1 \cap \theta_3 = \emptyset$ 0.208	$\theta_2 \cap \theta_3 = \emptyset$ 0.08	$\emptyset$ 0.064	θ_3 0.048
$m_3(I_t)$ 0.1	θ_1 0.052	θ_2 0.02	$\emptyset$ 0.016	I_t 0.012

Une étape finale de regroupement donne la table finale 2.4. On y retrouve l'objet dominant θ_1 qui est l'identité choisie par le décideur s'il se base sur la masse la plus élevée et que l'on rejete $\emptyset$. On présume bien entendu qu'il n'y a qu'une seule cible dans le champ de détection des capteurs.

2.3.2 Combinaison disjonctive

La combinaison disjonctive est une règle de combinaison élémentaire consistant à prendre les sommes de toutes les possibilités de disjonction entre tous les objets. En fait, la combinaison disjonctive de deux objets consiste à prendre le produit des masses des deux objets à combiner et à attribuer le résultat du produit à l'objet résultant de l'union des deux objets. Une fois cette opération réalisée, on regroupe les éléments

TAB. 2.4 – Résultats finaux de la combinaison de BOE 1, BOE 2, et BOE 3 par le biais de la règle conjonctive

	θ_1	θ_2	θ_3	?	I_t
m_{123}	0.372	0.02	0.048	0.548	0.012

communs. Cette règle de combinaison s'applique principalement lorsqu'il y a un conflit important entre les informations.

$$p(A) = m_1 \vee m_2 = \sum_{\substack{B \cup C = A \\ B, C \subseteq \Theta}} m_1(B)\, m_2(C) \qquad \forall A \subseteq \Theta \tag{2.5}$$

Exemple 2.3 Combinaison d'informations avec la règle disjonctive

Reprenant les informations fournies de l'exemple 2.1 regroupées au sein de la table 2.1, on procède encore une fois par la méthode dynamique. Comme dans l'exemple 2.2, la première étape consiste à faire le produit orthogonal entre les informations du BOE 1 *avec ceux du BOE* 2*. Selon l'équation (2.5) on obtient la table 2.5.*

TAB. 2.5 – Résultats d'une étape intermédiaire de la combinaison de BOE 1 et de BOE 2 par le biais de la règle disjonctive

	$m_1(\theta_1)$ 0.6	$m_1(\theta_2)$ 0.2	$m_1(I_t)$ 0.2
$m_2(\theta_1)$ 0.2	θ_1 0.12	$\theta_1 \cup \theta_2$ 0.04	I_t 0.04
$m_2(\theta_2)$ 0.2	$\theta_1 \cup \theta_2$ 0.12	θ_2 0.04	I_t 0.04
$m_2(I_t)$ 0.6	I_t 0.36	I_t 0.12	I_t 0.12

La seconde étape consiste à combiner les résultats regroupés de la combinaison de BOE 1 *et BOE* 2 *avec les informations de BOE* 3*. Il en résulte la table 2.6.*

L'étape finale de regroupement donne la table finale 2.7. On y retrouvera l'objet dominant I_t qui est l'objet choisi par le décideur sur la base de la plus grande masse.

TAB. 2.6 – Résultats de la combinaison de BOE 3 avec les résultats du tableau 2.5 par le biais de la règle disjonctive

	$m_{12}(\theta_1)$ 0.12	$m_{12}(\theta_2)$ 0.04	$m_{12}(\theta_1 \cup \theta_2)$ 0.16	$m_{12}(I_t)$ 0.68
$m_3(\theta_1)$ 0.5	θ_1 0.06	$\theta_1 \cup \theta_2$ 0.02	$\theta_1 \cup \theta_2$ 0.08	I_t 0.34
$m_3(\theta_3)$ 0.4	$\theta_1 \cup \theta_3$ 0.048	$\theta_2 \cup \theta_3$ 0.016	I_t 0.064	I_t 0.272
$m_3(I_t)$ 0.1	I_t 0.012	I_t 0.004	I_t 0.016	I_t 0.068

Cela signifie donc en fait qu'on n'est pas en mesure de prendre une décision sur l'identité de la cible dans le champ détecté par les capteurs, car I_t est l'ensemble de tous les objets, l'équivalent de l'ignorance totale.

TAB. 2.7 – Résultats finaux de la combinaison de BOE 1, BOE 2, et BOE 3 par le biais de la règle disjonctive

	θ_1	$\theta_1 \cup \theta_2$	$\theta_1 \cup \theta_3$	$\theta_2 \cup \theta_3$	I_t
m_{123}	0.06	0.10	0.048	0.016	0.776

2.3.3 Théorie de Dempster-Shafer

La théorie de Dempster-Shafer (DST) a été introduite par le biais des notions de probabilités supérieures et inférieures par Dempster[5]. Elle a ensuite été raffinée et présentée par Shafer par le biais des fonctions de croyances et de plausibilité[23]. Elle se veut une récente théorie mathématique permettant de traiter l'incertitude. À la base, on retrouve des masses de croyances au lieu de probabilités afin de déterminer, après des combinaisons d'évidences, le niveau de croyance, c'est à dire le niveau de certitude de l'occurrence d'un évènement.

Dans la théorie de l'évidence, les masses attribuées à l'ensemble A ne permettent pas de préciser à quel élément de A s'applique le niveau de croyance. Par contre, il y a la possibilité de délimiter les éléments auxquels la croyance s'applique. Ainsi, $\mathrm{Bel}(A) \leq \mathrm{Pr}(A) \leq \mathrm{Pl}(A)$, où Bel représente la fonction de croyance (*belief*), Pr représente la fonction de probabilité, et enfin Pl la fonction de plausibilité. On note également que ces deux fonctions, de croyance et de plausibilité, sont liées. En effet, l'équation (2.6) montre la relation entre les deux fonctions.

$$\mathrm{Pl}(A) = 1 - \mathrm{Bel}\left(\bar{A}\right) \tag{2.6}$$

Fonction de croyance

Tel que décrit par l'équation (2.7) la croyance de A, notée $\mathrm{Bel}(A)$, est représentée par la somme des masses des ensembles inclus dans A.

$$\mathrm{Bel}(A) = \sum_{B_i \subseteq A} m(B_i) \tag{2.7}$$

Fonction de plausibilité

La fonction de plausibilité, quant à elle, est représentée par la somme des masses des ensembles qui ont une intersection non nulle avec A.

$$\mathrm{Pl}(A) = \sum_{B_i \cap A} m(B_i) \tag{2.8}$$

Conflit conjonctif

Le conflit ($K_\cap$), également quelques fois désigné par contrainte[12] tel que défini au glossaire, survient lorsque des intersections impossibles (conjonctions contraintes) obtiennent une masse lors de la conjonction. Il est défini par :

$$K_\cap = \sum_{A \cap B = \emptyset} m_1(A)\, m_2(B) \qquad A, B \subseteq \Theta. \tag{2.9}$$

Exemple 2.4 Situation de conflit due à une contrainte

Avec les informations finales de l'exemple 2.2, plus particulièrement de la table 2.4, une contrainte est imposée sur toutes intersections des singletons. Cela oblige les masses attribuées à ces intersections à être maintenant associées à l'ensemble vide $\emptyset$. Il en résulte la table 2.8.

TAB. 2.8 – Combinaison de l'exemple 2.2 en considérant toutes conjonctions comme étant conflictuelles

	θ_1	θ_2	θ_3	$\theta_1 \cap \theta_2$	$\theta_1 \cap \theta_3$	$\theta_2 \cap \theta_3$	$\theta_1 \cap \theta_2 \cap \theta_3$	I_t	$\emptyset$
m_{123}	0.372	0.02	0.048	0.000	0.000	0.000	0.000	0.012	0.548

On remarque par l'exemple 2.4, qu'un conflit peut occasionner une accumulation de la masse vers l'ensemble $\emptyset$. Dans ce cas particulier, la majorité de la masse s'y retrouve. Cela a pour effet d'empêcher une décision fiable si θ_1 est choisi, ou même d'empêcher toutes possibilités de décision si on considère l'ensemble vide $\emptyset$. C'est principalement ce problème que tentent de résoudre les règles de combinaisons décrites maintenant.

Règle de combinaison de Dempster

La règle de combinaison de Dempster (DS) est une règle conjonctive normalisée fonctionnant sur l'ensemble de puissance 2^Θ. La théorie de Dempster-Shafer (DST) suppose que les sources d'évidence sont mathématiquement indépendantes. L'équation (2.10) décrit la règle de combinaison DS où $K_\cap$ représente le conflit conjonctif.

[12] C'est plus exactement une contrainte sur un objet ou ensemble qui lorsqu'il se voit attribuer une masse, constitue une masse faisant partie du conflit conjonctif tel que défini par l'équation 2.9.

$$(m_1 \oplus m_2)(C) = \frac{1}{1 - K_\cap} \sum_{A \cap B = C} m_1(A)\, m_2(B) \qquad \forall C \subseteq \Theta \tag{2.10}$$

En fait, on redistribue la masse associée à l'ensemble vide aux autres ensembles selon leur masse respective.

Exemple 2.5 Redistribution du conflit conjonctif avec la règle de Dempster

Considérant l'exemple 2.4, la règle de Dempster répartit $K_\cap$ proportionnellement suivant l'équation 2.10. La résultante est présentée dans la table 2.9.

TAB. 2.9 – Combinaison de l'exemple 2.2 avec redistribution de la masse conflictuelle suivant la règle de Dempster

	θ_1	θ_2	θ_3	$\theta_1 \cap \theta_2$	$\theta_1 \cap \theta_3$	$\theta_2 \cap \theta_3$	$\theta_1 \cap \theta_2 \cap \theta_3$	I_t	$\emptyset$
m_{123}	0.823	0.044	0.106	0.000	0.000	0.000	0.000	0.027	0.000

Particularité de la théorie de Dempster-Shafer

Pouvant raisonner dans l'incertain, la théorie de Dempster-Shafer est également en mesure de généraliser les théories de probabilités et de possibilités. La règle de Dempster est basée sur la conjonction, mais redistribue la masse de l'ensemble vide à tous les éléments focaux par une normalisation. C'est ainsi que la masse de l'ensemble vide redevient nulle, une fois la redistribution effectuée. La règle de combinaison de Dempster est commutative et associative. Cette dernière particularité la rend d'ailleurs très attrayante. Malheureusement, elle n'est pas en mesure d'être fonctionnelle lorsque le conflit conjonctif s'approche trop de l'unité[13].

Paradoxe sur la précision et le conflit

Une personne avertie constatera le paradoxe suivant : *plus les sources d'informations sont certaines*[14]*, même si imprécises, plus le conflit peut être élevé. En conséquence, le résultat de la fusion devient moins certain.*

[13]Il en est ainsi, car le facteur de normalisation tend vers une division par zéro à mesure que le conflit conjonctif tend vers l'unité.

[14]Donc, plus la masse attribuée à un ensemble est élevée.

2.3.4 Problèmes de la combinaison de Dempster

La règle de Dempster ne peut être utilisée lorsque le conflit conjonctif tend vers l'unité. La cause potentielle pourrait être un problème dans la modélisation même des données. La modélisation semble pourtant être rarement soulevée comme étant cause du problème dans la littérature consultée. Mahler, très récemment, dans [20] en fait toutefois la preuve mathématique.

Le comportement peut aussi s'avérer très sensible par moment. Aussi, cette combinaison exige une grande quantité de calculs par rapport aux méthodes classiques. Jusqu'à récemment, les solutions au conflit élevé étaient séparées en deux catégories. Celles prônant la famille de règles conjonctives, attribuent la faute du conflit sur la fiabilité des sources ; celles prônant la famille de règles disjonctives, attribuent plutôt la faute sur l'ignorance des sources non fiables. La section 2.4 présente une nouvelle solution n'appartenant à aucune de ces deux catégories.

2.3.5 Règle de combinaison de Smets

La règle de combinaison de Smets considère les sources d'informations comme étant fiables. Elle équivaut à une version non normalisée de la règle de Dempster-Shafer et permet une masse non-nulle à l'ensemble vide. Cela élimine la division par $1 - K_\cap$ de la règle de Dempster-Shafer. Smets considère que deux sources fiables causant un conflit, devrait soulever l'hypothèse d'un problème mal posé et recommande de ne pas redistribuer la masse conflictuelle. [27, 28]

La règle de combinaison de Smets suppose un univers ouvert[15].

$$m_s(\emptyset) = K_\cap = \sum_{\substack{X,Y \in 2^\Theta \\ X \cap Y = \emptyset}} m_1(X)\, m_2(Y) \tag{2.11}$$

$$m_s(A) = m_\cap(A) = \sum_{\substack{X,Y \in 2^\Theta \\ X \cap Y = A}} m_1(X)\, m_2(Y) \qquad \forall A \in 2^\Theta \setminus \{\emptyset\} \tag{2.12}$$

[15] signifiant que la solution peut ne pas se retrouver dans l'ensemble Θ

Exemple 2.6 Combinaison d'informations avec la règle de Smets

Avec les informations finales de l'exemple 2.2, plus particulièrement de la table 2.4, une contrainte sur toutes intersections est imposé. Cela oblige les masses attribuées à ces intersections à être maintenant associée à l'ensemble vide $\emptyset$. La règle est commutative et associative et donc l'ordre de combinaison n'est pas important. C'est la raison pour laquelle on peut appliquer les contraintes à la dernière étape de combinaison sans changer le résultat final, au lieu d'appliquer les contraintes après chaque itération. Il en résulte la table 2.10.

TAB. 2.10 – Combinaison de l'exemple 2.2 en considérant toutes conjonctions comme étant conflictuelle avec Smets

	θ_1	θ_2	θ_3	$\theta_1 \cap \theta_2$	$\theta_1 \cap \theta_3$	$\theta_2 \cap \theta_3$	$\theta_1 \cap \theta_2 \cap \theta_3$	I_t	$\emptyset$
m_S	0.372	0.02	0.048	0.000	0.000	0.000	0.000	0.012	0.548

2.3.6 Règle de combinaison de Yager

La règle de combinaison de Yager considère des sources d'informations non fiables. Cette règle propose l'ajout d'une hypothèse au cadre de discernement sur laquelle portera toute la masse conflictuelle que si I_t ne fait pas partie des éléments focaux de la combinaison conjonctive. Puisque cette règle considère le résultat non fiable en cas de conflit, elle redistribue la masse conflictuelle à l'ignorance [36, 37].

$$m_Y(\emptyset) = 0 \tag{2.13}$$

$$m_Y(A) = \sum_{\substack{X,Y \in 2^\Theta \\ X \cap Y = A}} m_1(X)\, m_2(Y) \qquad \forall A \in 2^\Theta \setminus (\{\emptyset\}, I_t) \tag{2.14}$$

$$m_Y(I_t) = m_\cap(I_t) + \sum_{\substack{X,Y \in 2^\Theta \\ X \cap Y = \emptyset}} m_1(X)\, m_2(Y) \qquad \forall A = I_t = \Theta \tag{2.15}$$

Exemple 2.7 Combinaison d'informations avec la règle de Yager

Avec les informations finales de l'exemple 2.2, (table 2.4), une contrainte sur toutes intersections est imposé. Cela oblige l'assignation d'une masse de 0.56 *à* I_t *en utilisant la méthode statique de combinaison. Il en résulte la table 2.11.*

TAB. 2.11 – Combinaison de l'exemple 2.2 avec redistribution de la masse conflictuelle suivant la règle de Yager statique

	θ_1	θ_2	θ_3	$\theta_1 \cap \theta_2$	$\theta_1 \cap \theta_3$	$\theta_2 \cap \theta_3$	$\theta_1 \cap \theta_2 \cap \theta_3$	I_t	$\emptyset$
m_Y	0.372	0.02	0.048	0.000	0.000	0.000	0.000	0.560	0.000

Suivant la méthode dynamique de combinaison, l'ordre de combinaison est important. En commençant par m_1 *et* m_2 *d'abord puis en ajoutant* m_3 *ensuite, on obtient les résultats présentés dans la table 2.12.*

TAB. 2.12 – Table 2.4 avec des contraintes sur intersections pour Yager dynamique

	θ_1	θ_2	θ_3	$\theta_1 \cap \theta_2$	$\theta_1 \cap \theta_3$	$\theta_2 \cap \theta_3$	$\theta_1 \cap \theta_2 \cap \theta_3$	I_t	$\emptyset$
m_Y	0.452	0.02	0.112	0.000	0.000	0.000	0.000	0.416	0.000

On constate qu'avec certaines règles de combinaison, il est possible que le résultat de la méthode dynamique diffère du résultat de la méthode statique. Suivant l'ordre d'arrivée des informations à combiner, une règle non associative provoque très certainement, comme dans l'exemple 2.7, des résultats différents.

2.3.7 Règle de combinaison de Dubois et Prade

La règle de combinaison de Dubois et Prade considère aussi des sources d'informations non fiables. Il est reconnu qu'elle est un juste équilibre entre précision et fiabilité. En fait, cette règle suppose qu'au moins une de deux sources est correcte lors d'un conflit et que les deux sont fiables en l'absence de conflit. Ainsi la solution doit se trouver soit dans la disjonction, soit dans la conjonction [10, 11]. Sans conflit, elle entraînera la combinaison conjonctive.

$$m_{DP}(\emptyset) = 0 \tag{2.16}$$

$$m_{DP}(A) = \sum_{\substack{X,Y \in 2^\Theta \\ X \cap Y = A \\ X \cap Y \neq \emptyset}} m_1(X)\, m_2(Y) + \sum_{\substack{X,Y \in 2^\Theta \\ X \cup Y = A \\ X \cap Y = \emptyset}} m_1(X)\, m_2(Y) \qquad \forall A \in 2^\Theta \backslash \emptyset \tag{2.17}$$

Exemple 2.8 Combinaison d'informations avec la règle de Dubois et Prade

Reprenant cette fois les données initiales de l'exemple 2.2 avec contrainte sur toutes intersections, on applique la règle de Dubois et Prade. À la première étape de la combinaison, les masses conflictuelles sont maintenant assignées aux unions. La table 2.13 présente le cas où des masses sont reportées vers la disjonction.

TAB. 2.13 – Combinaison de BOE 1 et de BOE 2 suivant la règle de Dubois et Prade

	$m_1(\theta_1)$ 0.6	$m_1(\theta_2)$ 0.2	$m_1(I_t)$ 0.2
$m_2(\theta_1)$ 0.2	$m_{DP}(\theta_1)$ 0.12	$m_{DP}(\theta_1 \cup \theta_2)$ 0.04	$m_{DP}(\theta_1)$ 0.04
$m_2(\theta_2)$ 0.2	$m_{DP}(\theta_1 \cup \theta_2)$ 0.12	$m_{DP}(\theta_2)$ 0.04	$m_{DP}(\theta_2)$ 0.04
$m_2(I_t)$ 0.6	$m_{DP}(\theta_1)$ 0.36	$m_{DP}(\theta_2)$ 0.12	$m_{DP}(I_t)$ 0.12

2.3.8 Règle de combinaison de Inagaki

La règle de combinaison de Inagaki est une règle généraliste qui est en mesure de permettre à l'utilisateur de choisir le degré de disjonction versus celui de conjonction selon une fonction de pondération [18]. On voit qu'avec un poids de pondération $w_m(A) = 1$, la règle devient équivalent à la combinaison de Dubois et Prade.

$$m_{Ina}(\emptyset) = 0 \qquad \forall A \in D^\Theta \backslash \{\emptyset\} \tag{2.18}$$

$$m_{Ina}(A) = m_\wedge(A) + w_m(A)\, m_\wedge(\emptyset) \qquad w_m(A) \in [0,1]\, \forall A \in D^\Theta \backslash \{\emptyset\} \tag{2.19}$$

2.3.9 Analyse comparative théorique de quelques règles classiques

Toutes les règles de combinaisons vues sont basées sur la règle conjonctive, outre la règle disjonctive seule. Les différences se situent au niveau de la répartition de la masse conflictuelle. Voici une récapitulation des différentes approches de redistribution.

- **règle conjonctive :** aucune.
- **règle disjonctive :** aucune.
- **règle de Demptser-Shafer :** proportionnellement aux masses des éléments focaux non conflictuels.
- **règle de Yager :** vers l'ignorance totale.
- **règle de Dubois et Prade :** vers l'ignorance partielle. (c.-à-d. les unions)
- **règle de Smets :** vers l'ensemble vide.
- **règle de Inagaki :** généralisation par le biais de pondérations des règles partageant en totalité ou en partie entre la conjonction et la disjonction la masse conflictuelle.

2.4 Nouvelle génération de règles de fusion

2.4.1 Cadre de raisonnement DSm

Le cadre de raisonnement DSm fut conçu conjointement par Jean Dezert et Florentin Smarandache[25]. Il constitue une nouvelle manière de représenter et fusionner les information incertaine. Il est défini comme étant en complémentarité avec le cadre de raisonnement de base afin de l'étendre à l'ensemble d'hyperpuissance D^{Θ}.

Avec le cadre de discernement $\Theta = \{\theta_1, \theta_2, \dots \theta_n\}$, l'ensemble de puissance 2^{Θ} ne comprend que des unions des singletons, les singletons, l'ensemble vide, ainsi que l'union totale.

L'ensemble d'hyperpuissance comprend en plus, les intersections, les unions d'intersections. Pour montrer sa construction, il vaut mieux se restreindre à un cas particulier avec $n = 3$.

$$D^{\Theta}\left(|\Theta|=3\right) \equiv \{\{\emptyset\},\{\theta_1\},\{\theta_2\},\{\theta_3\},\{\theta_1\cup\theta_2\},\{\theta_1\cup\theta_3\},\{\theta_2\cup\theta_3\},$$
$$\{\theta_1\cap\theta_2\},\{\theta_1\cap\theta_3\},\{\theta_2\cap\theta_3\},\{(\theta_1\cup\theta_2)\cap\theta_3\},\{(\theta_1\cup\theta_3)\cap\theta_2\},\{(\theta_2\cup\theta_3)\cap\theta_1\},$$
$$\{(\theta_1\cap\theta_2)\cup\theta_3\},(\{\theta_1\cap\theta_3)\cup\theta_2\},\{(\theta_2\cap\theta_3\cup\theta_1)\},\{\theta_1\cap\theta_2\cap\theta_3\},$$
$$\{\theta_1\cup\theta_2\cup\theta_3\},\{(\theta_1\cap\theta_2)\cup(\theta_1\cap\theta_3)\cup(\theta_2\cap\theta_3)\}\}$$

Cardinal du cadre de raisonnement DSm

Tel que vu ci-dessus, le cardinal du cadre de raisonnement DSm s'accroît en fonction du cardinal du cadre de discernement sur lequel il est basé. On voit dans [34] qu'un objet mathématique comparable est désigné d'*Antichaine*. Selon [34], une *antichaine* est également désignée par *système de Sperner* dans la littérature mathématique plus ancienne.

La détermination du cardinal des *antichaines* est également connue comme étant le problème de Dedekind et la séquence de ces cardinaux est quelques fois désignée comme étant la séquence de Dedekind. Voici une table de cette séquence de Dedekind en fonction du cardinal du cadre de discernement.

TAB. 2.14 – Sequence de Dedekind

Cardinal de Θ	0	1	2	3	4	5	6	7	8
Cardinal de D^{Θ}	1	2	5	19	167	7580	7828353	241×10^{10}	561×10^{20}

La section 3.3.5 (page 60) revoit cette séquence de cardinal du point de vue de la complexité de la génération de l'ensemble d'hyperpuissance.

2.4.2 Précision sur la terminologie employée

ALLÉGEANCE

L'allégeance d'une cible est son état de fidélité à un pays. La cible pourra donc être ami, ennemi, neutre, suspect ou présumé ami. On désignera ce type d'allégeance qui peut être répartis parmi ces cinq choix l'*allégeance stanag*. On considère toutefois que les systèmes ESM fournissent seulement l'information sur l'allégeance d'une cible parmi

les suivants : ami, ennemi, ou neutre. Ce type d'allégeance qui peut être répartis parmi ces trois choix sera désigné par l'*allégeance esm*. Les systèmes ESM donnent également un taux de certitude par rapport à cette information d'allégeance.

STANAG

Abréviation de Standardization Agreement, les STANAG définissent des procédures, termes et conditions adoptées par les membres de l'OTAN.

STANAG 4162

Le processus de combinaison de données d'identification (IDPC), un processus de fusion de donnée générique, ainsi que les caractéristiques techniques de systèmes d'identification de l'OTAN sont définis au sein du STANAG 4162. Le STANAG 4162, qui définit le processus d'identification, fait référence au STANAG 1241 quant à la catégorisation des identifications, entre autres, au niveau de l'allégeance des cibles identifiées.

STANAG 1241

Le STANAG 1241 définit la description de la structure de l'identification pour usage tactique. On y définit entre autres, la structure ou répartition de l'allégeance possible des cibles. L'allégeance possible d'une cible peut être faite parmi les cinq choix suivants : ami, présumé ami, neutre, suspect, et puis ennemi.

2.4.3 Théorie de Dezert-Smarandache

La DSmT fonctionne avec l'ensemble d'hyperpuissance. Ainsi, la DSmT est en mesure de fonctionner correctement non seulement avec les unions mais également avec les intersections. De plus, la DSmT règle le problème de redistribution de masses hautement conflictuelles survenant sous la théorie de l'évidence. La DSmT comporte deux règles de combinaison : la version classique et la version hybride.

Types de contraintes possibles

Une contrainte d'intégrité d'un ensemble U est en fait une impossibilité de considérer une attribution de masse à cet ensemble. La masse de l'ensemble U est alors assignée à l'ensemble vide $\emptyset$. L'ensemble U est alors considéré comme étant un conflit et les règles de combinaisons doivent agir en conséquence. Voici brièvement une revue de quelques types de contraintes d'intégrités présentés dans [25].

Contrainte d'exclusion : La contrainte d'exclusion a lieu lorsque l'on considère une conjonction d'éléments $\theta_i, \dots \theta_k$ comme étant impossible à obtenir. Ainsi, on aurait l'équivalence $\theta_i \cap \dots \cap \dots \cap \theta_k \equiv \emptyset$ dans ce cas particulier.

Contrainte d'inexistence : La contrainte d'inexistence a lieu quant à elle, lorsque l'on considère une disjonction d'éléments $\theta_i, \dots \theta_k$ comme étant impossible à obtenir. Ainsi, on aurait l'équivalence $\theta_i \cup \dots \cup \dots \cup \theta_k \equiv \emptyset$ dans ce cas particulier.

Contrainte d'intégrité hybride : La contrainte d'intégrité hybride est en fait une mixture des deux types de contrainte d'intégrité précédentes. Ainsi, on peut par exemple avoir une contrainte sur $(\theta_i \cap \theta_j) \cup \theta_k$ ou tout autre élément comportant à la fois des intersections et des unions.

Héritage de contraintes : Une règle/propriété importante définie dans [25] stipule qu'une contrainte sur un ensemble A oblige tous les sous-ensembles B, tels que $B \subseteq A$ à être également contraint. Il y a donc héritage de la contrainte d'un ensemble vers les sous-ensembles.

Ambiguïté sur la contrainte d'inexistence

Considérant la situation où l'on veut contraindre[16] la prise de décision parmi un choix θ_i ou θ_j. On souhaite donc seulement mettre une contrainte sur l'union $\theta_i \cup \theta_j$. Un problème survient alors lorsque l'on considère fixe la propriété sur l'héritage de contraintes. Dans ce contexte, on définit donc la **contrainte d'intégrité d'inexistence stricte**, où seulement l'union est affectée et non ses sous-ensembles.

Particularité 1 : Lorsque la contrainte porte sur un ensemble A, les ensembles conjonctifs qui l'incluent, sont également contraints.

[16] Cela revient à dire : rendre impossible l'occurence de.

Particularité 2 : Lorsque la contrainte porte sur un ensemble A, un ensemble mixte à la fois conjonctif et disjonctif, dont la partie conjonctive inclue A, $(A \cap B) \cup C$ par exemple, alors $A \cap B$ devient également contraint, il ne restera alors que C pour cet exemple spécifique.

Particularité 3 : Lorsque la contrainte porte sur un ensemble A, alors la conjonction avec A est retirée de tous les ensembles disjonctifs qui comprennent A, (donc $A \cup B \cup C$ devient $B \cup C$ seulement).

Modèle de répartition de l'information

Modèle de Shafer $[\mathcal{M}^0(\Theta)]$ Le modèle de Shafer est un modèle où tout les singletons doivent être exclusifs et distincts. Il correspond à l'ensemble de puissance 2^Θ. Il s'agit donc du modèle où seules les combinaisons de singletons avec l'opérateur disjonctif sont considérées, toutes conjonctions étant exclues.

Modèle libre $[\mathcal{M}^f(\Theta)]$ Dans le modèle libre, par opposition au modèle de Shafer, on considère cette fois comme étant possible que les objets soient flous, vagues, imprécis, et donc, non-exclusifs et distincts. Cela signifie que les conjonctions sont également possibles et on se retrouve avec un modèle qui correspond à l'ensemble d'hyperpuissance D^Θau complet, sans aucune contrainte. Ainsi, toutes combinaisons des singletons par des unions et/ou des intersections sont possibles dans le cadre de ce modèle.

Modèle hybride $[\mathcal{M}(\Theta)]$ Partant du modèle libre $\mathcal{M}^f(\Theta)$, on considère maintenant possible la présence de contraintes. On se retrouve avec un modèle hybride où certains ensembles ne sont pas possibles.

Règle de combinaison classique (DSmC)

La règle de combinaison classique de la DSmT permet une combinaison équivalente à la DST mais fonctionnant sous l'environnement D^Θ.

$$m(C) = \sum_{A \cap B = C} m_1(A)\, m_2(B) \qquad A, B \in D^\Theta, \forall C \in D^\Theta \tag{2.20}$$

Règle de combinaison hybride (DSmH)

La règle de combinaison hybride de la DSmT permet la gestion et le fonctionnement malgré divers types de contraintes, ce que la version classique n'est pas en mesure de faire.

$$m_{M(\Theta)}(A) = \phi(A)\left[S_1(A) + S_2(A) + S_3(A)\right] \qquad \forall A \in D^\Theta \tag{2.21}$$

$$S_1(A) = \sum_{X_1 \cap X_2 = A} m_1(X_1)\, m_2(X_2) \qquad \forall X_1, X_2 \in D^\Theta \tag{2.22}$$

$$S_2(A) = \sum_{[(u(X_1)\cup u(X_2))=A]\vee[((u(X_1)\cup u(X_2))\in\emptyset)\wedge(A=I_t)]} m_1(X_1)\, m_2(X_2) \qquad \forall X_1, X_2 \in \emptyset \tag{2.23}$$

$$S_3(A) = \sum_{X_1 \cup X_2 = A} m_1(X_1)\, m_2(X_2) \qquad \forall X_1, X_2 \in D^\Theta \quad \text{et} \quad X_1 \cap X_2 \in \emptyset \tag{2.24}$$

La fonction $\phi(A)$ de l'équation (2.21) est une fonction binaire égale à zéro pour les ensembles A vides ou impossibles, autrement elle vaut l'unité. Dans l'équation (2.23), $u(X)$ représente l'union de tous les objets de l'ensemble X. L'équation (2.24) indique que la masse est attribuée à l'union de tous les objets des ensembles X_1 et X_2, si l'intersection est une contrainte. Quant à l'équation (2.23), elle indique que si l'union des objets est aussi une contrainte, alors la masse est soit assignée à l'union de tous les singletons qui forment les objets, soit à l'ignorance totale I_t.

Afin de mieux comprendre la DSmH, on a proposé dans [6] de voir cette règle de combinaison comme une procédure. La table 2.15[17] montre cette procédure qui détaille clairement comment combiner avec la DSmH. Évidemment, la procédure est équivalente à la version mathématique, mais permet une meilleure compréhension ainsi qu'une implémentation plus aisée.

[17] S_1, S_2 et S_3, de la table, correspondent aux S_1, S_2 et S_3 de l'équation 2.21

TAB. 2.15 – Procédure à appliquer de la DSmH à chaque paire d'ensemble (X_1, X_2)

Étape S_1 $(X_1 \cap X_2)$	Si $(X_1 \cap X_2)$ est une contrainte, alors continuer à S_3, sinon, $m_1(X_1)\, m_2(X_2)$ s'ajoute à la masse de $A = (X_1 \cap X_2)$.
Étape S_3 $(X_1 \cup X_2)$	Si $(X_1 \cup X_2)$ est une contrainte, alors continuer à S_2, sinon, $m_1(X_1)\, m_2(X_2)$ s'ajoute à la masse de $A = (X_1 \cup X_2)$.
Étape S_2 $(u(X_1) \cup u(X_2))$	Si $(u(X_1) \cup u(X_2))$ est une contrainte, alors ajouter sa masse à I_t, sinon, $m_1(X_1)\, m_2(X_2)$ est ajoutée à la masse $A = (u(X_1) \cup u(X_2))$.

Exemple 2.9 Combinaison d'informations avec la DSmH

À partir des informations de l'exemple 2.2, on procède par la méthode dynamique. Pour cet exemple, les trois ensembles $\theta_1 \cap \theta_2$, $\theta_1 \cap \theta_3$, et $\theta_1 \cup \theta_3$ sont considérés comme étant des contraintes. Où $\theta_1 \cup \theta_3$ est plus spécifiquement une contrainte d'intégrité d'inexistence stricte.

En procédant tel que décrit par la table 2.15, on commence en premier lieu à une conjonction de BOE 1 et BOE 2. À l'occurrence de la conjonction contrainte $\theta_1 \cap \theta_2$, on considère plutôt la disjonction des intervenants ayant mené à cet ensemble contraint. Ainsi, on assigne la masse à $\theta_1 \cup \theta_2$ comme indiqué à la table 2.16.

TAB. 2.16 – Combinaison de BOE 1 et de BOE 2 suivant la DSmH

	$m_1(\theta_1)$ 0.6	$m_1(\theta_2)$ 0.2	$m_1(I_t)$ 0.2
$m_2(\theta_1)$ 0.2	θ_1 0.12	$\theta_1 \cup \theta_2$ 0.04	θ_1 0.04
$m_2(\theta_2)$ 0.2	$\theta_1 \cup \theta_2$ 0.12	θ_2 0.04	θ_2 0.04
$m_2(I_t)$ 0.6	θ_1 0.36	θ_2 0.12	I_t 0.12

La seconde étape consiste à combiner les résultats de la combinaison de BOE 1 et BOE 2 avec les informations de BOE 3. Il en résulte la table 2.17. Cette étape marque également quelques particularités, dont la plus intéressante qui se trouve en la combinaison $m_{12}(\theta_1) \wedge m_3(\theta_3)$. Cette combinaison, normalement donne $\theta_1 \cap \theta_3$, mais

à cause de la contrainte sur cet ensemble, la DSmH propose d'assigner la masse à $\theta_1 \cup \theta_3$*. Par contre, cet ensemble-là est également contraint. Suivant la table 2.15, plus particulièrement à l'étape S2, on doit assigner la masse correspondante à l'union des éléments des sous-ensembles des intervenants ayant mené au conflit. Or, le conflit est également sur* $(u(\theta_1) \cup u(\theta_3)) = \theta_1 \cup \theta_3$*. On assigne alors la masse vers l'ignorance totale* I_t*. On présente, à même la table 2.17 le résultat qui considère les particularités présentées ci-haut.*

TAB. 2.17 – Combinaison de BOE 3 avec les résultats du tableau 2.16 suivant la DSmH

	$m_{12}(\theta_1)$ 0.52	$m_{12}(\theta_2)$ 0.20	$m_{12}(\theta_1 \cup \theta_2)$ 0.16	$m_{12}(I_t)$ 0.12
$m_3(\theta_1)$ 0.5	θ_1 0.26	$\theta_1 \cup \theta_2$ 0.10	θ_1 0.08	θ_1 0.06
$m_3(\theta_3)$ 0.4	I_t 0.208	$\theta_2 \cap \theta_3$ 0.08	$(\theta_1 \cup \theta_2) \cap \theta_3$ 0.064	θ_3 0.048
$m_3(I_t)$ 0.1	θ_1 0.052	θ_2 0.02	$\theta_1 \cup \theta_2$ 0.016	I_t 0.012

Une étape finale de regroupement donne la table finale 2.18. On y retrouve l'objet dominant θ_1 *qui est l'objet choisi par le décideur sur la base de la plus grande masse. Notez que la masse de* 0.22 *que l'on aurait attribué à* I_t *se retrouve en fin de compte sur* θ_2 *étant donnée la contrainte d'intégrité d'inexistance stricte qui affecte* $\theta_1 \cup \theta_3$*.*

TAB. 2.18 – Résultats finaux de la combinaison de BOE 1, BOE 2, et BOE 3 suivant la DSmH avec les contraintes imposées.

	θ_1	θ_2	θ_3	$\theta_1 \cup \theta_2$	$\theta_2 \cap \theta_3$	$(\theta_1 \cup \theta_2) \cap \theta_3$
m_{123}	0.452	0.24	0.048	0.116	0.08	0.064

Règle de combinaison hybride généralisée

La DSmH telle que présentée par les équations (2.21) à (2.24) n'est valable que dans les situations impliquant deux sources. Dans la situation où l'on a k sources à combiner, on procède avec les équations suivantes :

$$m_{M(\Theta)}(A) = \phi(A)\left[S_1(A) + S_2(A) + S_3(A)\right] \tag{2.25}$$

$$S_1(A) = \sum_{X_1 \cap X_2 \cap \ldots \cap X_k = A} \prod_{i=1}^{k} m_i(X_i) \qquad \forall X_1, X_2, \ldots, X_k \in D^\Theta \tag{2.26}$$

$$S_2(A) = \sum_{[\mathcal{U}=\mathcal{A}] \vee [(\mathcal{U} \in \emptyset) \wedge (A = I_t)]} \prod_{i=1}^{k} m_i(X_i) \qquad \forall X_1, X_2, \ldots, X_k \in \emptyset \tag{2.27}$$

$$S_3(A) = \sum_{X_1 \cup X_2 \cup \ldots \cup X_k = A} \prod_{i=1}^{k} m_i(X_i) \qquad \forall X_1, X_2, \ldots, X_k \in D^\Theta \qquad X_1 \cap X_2 \cap \ldots \cap X_k \in \emptyset \tag{2.28}$$

2.4.4 Règle de combinaison à redistribution proportionnelle

La famille de règles de combinaison à Redistribution Proportionnelle du Conflit (PCR) opère dans l'optique d'une répartition proportionnelle du conflit partielle aux éléments focaux impliqués dans la génération du conflit ou de l'élément contraint. Le principe consiste à calculer la fonction de masse pour une combinaison conjonctive, puis d'en évaluer les masses conflictuelles pour ensuite redistribuer cette masse conflictuelle, totalement ou partiellement proportionnellement aux ensembles à masse non nulle impliqués dans la combinaison.

Seulement la 5e version de la famille de règles PCR est explorée dans la suite, car elle est celle qui donne les meilleurs résultats selon diverses études [13, 26]. La règle PCR peut fonctionner autant sous la DST que sous la DSmT (pour les trois types de modèles explorés à la section 2.4.3) avec des cas statiques et dynamiques. Pour deux sources d'informations à combiner, on a l'équation suivante :

$$m_{PCR}(A) = m_{\wedge}(A) + \sum_{B \in D^{\Theta} \setminus \{A\}} \sum_{A \cap B = \emptyset} \left[\frac{m_1(A)^2 m_2(B)}{m_1(A) + m_2(B)} + \frac{m_2(A)^2 m_1(B)}{m_2(A) + m_1(B)} \right] \quad (2.29)$$

Exemple 2.10 Combinaison d'informations avec la règle PCR

À partir de la table 2.3 et considérant cette fois l'ensemble $\theta_1 \cap \theta_2 \cap \theta_3$ comme seule contrainte du problème, les particularités suivantes sont observées. Sans contrainte, cette règle se résume à la règle conjonctive de l'équation (2.29). On part donc de la seconde étape de la combinaison conjonctive en considérant que l'intersection totale est la seule contrainte.

On arrive à une situation où l'on a $m_{12}(\theta_1 \cap \theta_2) \wedge m_3(\theta_3) = m_{123}(\theta_1 \cap \theta_2 \cap \theta_3)$. Elle correspond à la seule contrainte du cas présent. La PCR, comme expliqué ci-haut, remet aux intervenants du conflit de façon proportionnelle, la masse conflictuelle qu'ils ont causée. Ainsi, on obtient par la répartition de la masse conflictuelle de 0.064*, la répartition suivante : à $m_{123}(\theta_1 \cap \theta_2) = 0.16(0.064/(0.16 + 0.4)) = 0.0183$ et à $m_{123}(\theta_3) = 0.4(0.064/(0.16 + 0.4)) = 0.0457$.*

On note que cette méthode revient à ne pas opérer sur les intervenants d'un conflit et leur laisser conserver une masse résiduelle proportionnelle à leur masse par rapport à la somme de la masse des deux intervenants du conflit.

TAB. 2.19 – Seconde étape de combinaison suivant la règle PCR

	$m_{12}(\theta_1)$ 0.52	$m_{12}(\theta_2)$ 0.20	$m_{12}(\theta_1 \cap \theta_2)$ 0.16	$m_{12}(I_t)$ 0.12
$m_3(\theta_1)$ 0.5	θ_1 0.26	$\theta_1 \cap \theta_2$ 0.10	$\theta_1 \cap \theta_2$ 0.08	θ_1 0.06
$m_3(\theta_3)$ 0.4	$\theta_1 \cap \theta_3$ 0.208	$\theta_2 \cap \theta_3$ 0.08	$\theta_1 \cap \theta_2$ θ_3 0.0183 0.0457	θ_3 0.048
$m_3(I_t)$ 0.1	θ_1 0.052	θ_2 0.02	$\theta_1 \cap \theta_2$ 0.016	I_t 0.012

2.4.5 Règle de combinaison adaptative

La Règle de Combinaison Adaptative (ACR) s'adapte pour donner plus de poids au résultat des combinaisons conjonctives ou disjonctives selon une pondération basée sur le conflit conjonctif. Selon [13], la règle est fonctionnelle et définie sur l'ensemble de puissance (2^Θ) seulement. La règle ne permet pas d'attribuer une masse à l'ensemble vide et procède à la combinaison par l'équation suivante :

$$m_{ACR}(A) = \alpha(K_\cap)m_\vee(A) + \beta(K_\cap)m_\wedge(A) \tag{2.30}$$

Les fonctions α et β dépendent du conflit $K_\cap$, où $K_\cap = m_\wedge(\emptyset) \in [0, 1]$, pour $(\alpha, \beta) \in [0, \infty]$. Les fonctions de pondération α et β doivent être choisies telles que m_{ACR} donne plus de poids à la partie disjonctive à mesure que $K_\cap$ tend vers 1, car à cette limite on présume qu'au moins une source est non fiable. À l'opposé, lorsque $K_\cap$ tend vers 0, (c.-à-d. aucun conflit ou conflit négligeable), on souhaite plutôt une m_{ACR} au comportement conjonctif.

Les fonctions de pondération α et β doivent au moins se prêter aux deux propriétés suivantes :
- $\beta(K_\cap)$ est décroissant de $\beta(0) = 1$ à $\beta(1) = 0$,
- $\alpha(K_\cap) = 1 - (1 - K_\cap)\beta(K_\cap)$.

ACR symétrique

La version symétrique de la règle ACR, la SACR, est une version de l'ACR qui comporte quelques propriétés relativement intéressantes. On obtient une règle ACR symétrique avec la formulation suivante :

$$m_{SACR}(A) = \alpha_0(K_\cap)m_\vee(A) + \beta_0(K_\cap)m_\wedge(A) \tag{2.31}$$

$$\alpha_0(K_\cap) = \frac{K_\cap}{1 - K_\cap + K_\cap^2} \tag{2.32}$$

$$\beta_0(K_\cap) = \frac{1 - K_\cap}{1 - K_\cap + K_\cap^2} \tag{2.33}$$

Exemple 2.11 Combinaison d'informations avec la règle SACR

À partir des tables 2.2, 2.5 provenant des exemples 2.2, 2.3, on utilise la règle ACR avec une contrainte sur $\theta_1 \cap \theta_2$ seulement. À partir des données de la table 2.2, on établit rapidement le conflit à $K_\cap = 0.16$. La seconde étape de la combinaison consiste à évaluer la valeur des fonctions de pondération des conjonctions et disjonctions, soient de α et de β. Selon les définitions, on obtient $\alpha = 0.1848$ et $\beta = 0.9704$. La troisième étape consiste à multiplier le pondérant de la conjonction, soit β, à l'ensemble des valeurs de la table 2.2 en ayant pris soin de retirer les masses attribuées aux intersections en conflit ; ainsi que de multiplier le pondérant de la disjonction, soit α, à l'ensemble des valeurs de la table 2.5. La dernière étape consiste à additionner les valeurs de deux tables de conjonction et de disjonction une fois pondérées par β et α respectivement. On retrouve à la table 2.20, les résultats finaux de la combinaison ACR.

TAB. 2.20 – Résultats finaux de la combinaison avec la règle SACR

	θ_1	θ_2	$\theta_1 \cup \theta_2$	I_t
m_{12}	0.5268	0.2015	0.0296	0.2421

2.5 Transformation Pignistique

2.5.1 Cardinalité

Avant d'introduire la cardinalité sous DSmT, on revoit brièvement ce qu'est le cardinal d'un ensemble. Le cardinal est un concept permettant de quantifier la dimension d'un ensemble. Il informe sur la taille de ce dernier. Plus particulièrement, le cardinal d'un ensemble[18] fini représente le nombre d'éléments qu'il comporte. Par exemple, on a donc : $|\{a, b, c, d\}| = 4$.

[18]Plus spécifiquement d'un ensemble classique, c.-à-d. où les éléments sont exclusifs.

2.5.2 Cardinalité dans DSm

La cardinalité dans DSm (ou cardinal DSm) d'un ensemble A, que l'on dénote $C_{\mathcal{M}}(A)$ comptabilise en fait le nombre total de partitions. Chacune des partitions possède un poids numérique unitaire. Ce poids identique pour tous les rend ainsi tous égaux. Le cardinal DSm sert dans les équations de transformation pignistique afin de distribuer la masse d'un ensemble A parmi toutes les partitions B, où $B \subseteq A$. Plus d'informations et de propriétés sont définies au chapitre 3 de [25].

Exemple 2.12 Cardinalité dans DSm

Le cardinal DSm d'un ensemble A, $C_{\mathcal{M}}(A)$, correspond au nombre d'ensembles et de sous-ensembles compris ou égal à A. Nous avons évalué les cardinaux DSm pour différents ensembles présentés à la table 2.21. Cette table permet de mieux visualiser la cardinalité dans DSm. Elle contient la valeur numérique du cardinal DSm des différents ensembles sous D^{Θ}, où $|\Theta| = 3$. Les différentes colonnes présentent les différentes valeurs de cardinalité DSm pour différents cas de modèles hybrides.

2.5.3 Transformation Pignistique Classique

La Transformation Pignistique Classique (TPC) est la transformation mathématique permettant de passer d'un modèle de Shafer à un modèle probabiliste. À la base on souhaite une conversion d'une fonction de croyance à une fonction de probabilité. Les détails se retrouvent au chapitre 7 de [25]. L'équation (2.34) permet de faire la conversion :

$$P\{A\} = \sum_{X \in 2^{\Theta}} \frac{|X \cap A|\, m(X)}{|X|} \tag{2.34}$$

On utilise[19] ensuite $\max(P\{A\})$ pour prendre une décision, car la fonction de plausibilité s'avère parfois trop optimiste et la fonction de croyance, trop pessimiste.

[19] $\max(BetP)$ est la principale notation utilisée dans la littérature pour cela.

TAB. 2.21 – Cardinaux DSm sous différents modèles hybrides de D^{Θ}avec $|\Theta| = 3$

	Contraintes			
	$\emptyset$	θ_3	$\theta_1 \cap \theta_3$	$\theta_1 \cap \theta_2 \cap \theta_3$
$\emptyset$	0	0	0	0
θ_1	4	2	2	3
θ_2	4	2	3	3
θ_3	4	0	2	3
$\theta_1 \cap \theta_2$	2	1	1	1
$\theta_1 \cap \theta_3$	2	0	1	1
$\theta_2 \cap \theta_3$	2	1	1	1
$\theta_1 \cap \theta_2 \cap \theta_3$	1	0	0	0
$\theta_1 \cup \theta_2$	6	3	4	5
$\theta_1 \cup \theta_3$	6	2	4	5
$\theta_2 \cup \theta_3$	6	2	4	5
$\theta_1 \cup \theta_2 \cup \theta_3$	7	3	5	6
$(\theta_1 \cap \theta_2) \cup \theta_3$	5	1	3	4
$(\theta_1 \cap \theta_3) \cup \theta_2$	5	2	3	4
$(\theta_2 \cap \theta_3) \cup \theta_1$	5	2	3	4
$(\theta_1 \cup \theta_2) \cap \theta_3$	3	0	1	2
$(\theta_1 \cup \theta_3) \cap \theta_2$	3	1	2	2
$(\theta_2 \cup \theta_3) \cap \theta_1$	3	1	1	2
$(\theta_2 \cap \theta_3) \cup (\theta_2 \cap \theta_3) \cup (\theta_2 \cap \theta_3)$	4	1	2	3

2.5.4 Transformation Pignistique Généralisée

La transformation mathématique qui permet de passer d'un modèle de représentation à fonction de masses sous D^{Θ} à un modèle probabiliste est désignée comme étant la Transformation Pignistique Généralisée (TPG). L'équation (2.35) définit l'opérateur de transformation :

$$P\{A\} = \sum_{X \in D^{\Theta}} \frac{C_{\mathcal{M}}(X \cap A)\, m(X)}{C_{\mathcal{M}}(X)} \tag{2.35}$$

où $C_{\mathcal{M}}$ correspond au cardinal DSm. Le chapitre 7 de [25] discute plus amplement du concept en donnant des propriétés et des exemples. Ces derniers démontrent que $P\{A\}$ est vraiment une probabilité mathématique.

2.6 Conclusions

Le présent chapitre a consisté en une revue de la littérature, des théories, concepts et outils mathématiques nécessaires à l'atteinte des objectifs définis à la section 1.2. Il comprend, entre autres, en plus des définitions de base (type d'incertitude, type de fusion, fonction de masse, méthode de combinaisons, conflit), une revue de règles de combinaisons classiques (règle conjonctive, disjonctive, Dempster-Shafer, Smets, Yager, Dubois et Prade, Inagaki), une description de la théorie de Dezert-Smarandache et ses particularités (cadre de raisonnement DSm, contrainte d'exclusion, contrainte d'inexistence, contrainte d'intégrité hybride, héritage de contraintes, modèle de répartition de l'information, règles de combinaison), une revue des nouvelles générations de règles de combinaisons sous la théorie de Dezert-Smarandache (PCR, ACR, SACR), et enfin, une revue de la transformation pignistique et de la cardinalité dans DSm. Dans l'optique d'atteinte des objectifs principaux, on présente au prochain chapitre ce que l'on a fait pour adapter les règles classiques pour fonctionner sous DSmT, établir une base pour comparer les règles sous la DSmT pour des données ESM avec le STANAG 1241[30], puis transposer en algorithme les diverses équations afin de les implanter.

Chapitre 3

Application du Stanag 1241 sous DSmT

Il n'est pas certain que tout soit incertain. – Blaise Pascal

3.1 Méthodologie de comparaison

Comme il se doit dans tout travail comportant analyses et comparaisons, il faut établir une base commune pour l'évaluation des performances selon diverses règles de combinaison d'informations dans différents cadres de raisonnement. Dans cette optique, on a décidé d'établir la comparaison entre les règles dans un même cadre de raisonnement autant que possible. D'autre part, parmi les divers éléments pouvant influencer le comportement des règles à l'étude, il y a les informations livrées en entrée et les contraintes. Tout en générant des modèles généraux d'informations et de contraintes assez aléatoires, les essais sont choisis en respectant le modèle de fonctionnement des ESM. Ces derniers sont à la base du contexte de ce travail de recherche.
La décision prise est ce qui compte de prime abord. On s'attend à ce que les règles de combinaison aient des qualités d'efficacité[1], de précision[2], de justesse[3] supérieures ou égales à celles qu'offre le raisonnement humain ainsi que la règle du vote majoritaire.

3.1.1 Cadre de raisonnement

Transposer les règles de combinaison dans le même espace de raisonnement sans perte est une chose essentielle pour établir une base de comparaison juste. Heureusement, le cadre de raisonnement DSmT utilise l'ensemble D^Θ qui est plus général que 2^Θ avec lequel travaille la théorie de l'évidence (DST). Ainsi, les règles pour lesquelles une transposition est possible, c'est-à-dire que le transfert vers D^Θ n'est pas contradictoire avec leur définition, mais surtout n'entachent pas leurs comportements, cette transposition vers D^Θ a été réalisée.

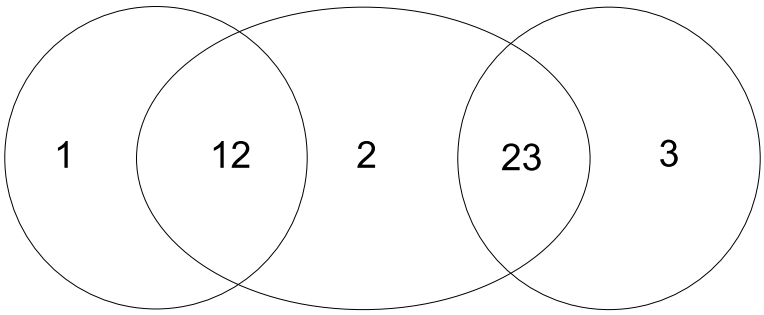
Le modèle d'allégeance de cible dans le standard de l'OTAN (identifié par STANAG 1241) répartit l'information d'une façon bien spécifique pour laquelle le cadre de raisonnement d'hyperpuissance D^Θ avec un modèle hybride s'avère intéressant.

[1] Capacité à prendre une décision rapidement.
[2] Capacité d'identification avec un bon niveau de certitude.
[3] Capacité à rendre la bonne décision.

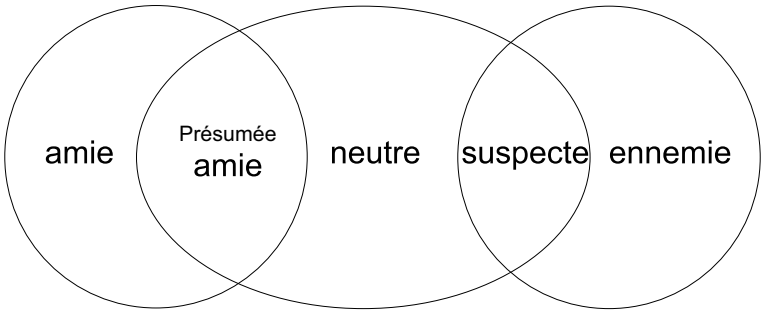
3.1.2 Conjonction du STANAG 1241 avec l'ensemble d'hyperpuissance

Le STANAG 1241 peut répartir les possibilités d'allégeance dans les cinq ensembles suivants : ami, neutre, ennemi, présumé ami, et suspect. La répartition des ensembles d'allégeance du STANAG 1241 peut être faite différemment selon le niveau de sécurité ou d'alerte, d'un pays. On retrouve deux types de répartition différents montrés aux figures[4] 3.1.1 et 3.2.1.

Le figures 3.1.1 et 3.1.2 réfèrent à la situation d'un pays en état de paix, puis les figures 3.2.1 et 3.2.2, réfèrent à celle d'un pays en état d'urgence.



3.1.1: Cas avec références numériques



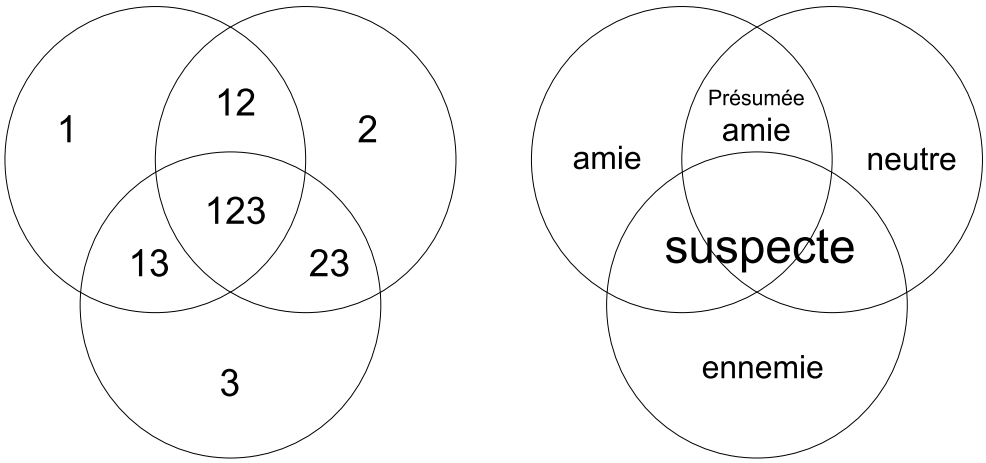
3.1.2: Cas avec références au STANAG

FIG. 3.1 – Cas d'un pays en état de paix

La correspondance vers un ensemble d'hyperpuissance se réalise aisément à partir de ces figures.

La figure 3.1.1 correspond à la figure 3.1.2, où on considère $ami = \{\theta_1 - \theta_1 \cap \theta_2\}$, $neutre = \{\theta_2 - \theta_1 \cap \theta_2 - \theta_2 \cap \theta_3\}$, $ennemi = \{\theta_3 - \theta_2 \cap \theta_3\}$, $présumé\ ami = \{\theta_1 \cap \theta_2\}$, $suspect = \{\theta_2 \cap \theta_3\}$. La notation $\theta_1 - \theta_1 \cap \theta_2$ signifie que θ_1 est amputé de l'intersection $\theta_1 \cap \theta_2$.

[4]La section 2.2.5 explique la notation de ces figures.



3.2.1: Cas avec références numériques 3.2.2: Cas avec références au STANAG

FIG. 3.2 – Cas d'un pays en état d'urgence

Quant à la figure 3.2.1, elle correspond à la figure 3.2.2, où on considère[5] $ami = \{1\}$, $neutre = \{2\}$, $ennemi = \{3\}$, $présumé\ ami = \{12\}$, $suspect = \{13, 23, 123\}$. On voit ainsi que l'ensemble d'hyperpuissance est en mesure de représenter naturellement le STANAG 1241.

3.1.3 Distinction à faire entre l'*allégeance stanag*, l'*allégeance esm*, et l'allégeance

L'*allégeance esm* est représentée à la figure 3.3, se définit comme à la section 2.4.2 et où les ensembles sont répartis et construis dans le cadre de raisonnement de base (voir section 2.2.2) en 2^{Θ}.

L'*allégeance stanag* est représentée à la figure 3.3, se définit comme à la section 2.4.2 et où les ensembles sont construis dans le cadre de raisonnement DSm (voir section 2.4.1) à la différence que les ensembles sont exclusifs. Ainsi, l'expression $\theta_1 \cap \theta_2 \subseteq \theta_1$ est fausse. Il en va de même pour tous les ensembles du cadre de raisonnement qui sont tous distincts et indépendants. Donc, dans le cas de l'*allégeance stanag*, ami= θ_1 exclue $\theta_1 \cap \theta_2$, car $\theta_1 \cap \theta_2$ = présumé ami. D'ailleurs, $\theta_1 \cap \theta_2$ exclue $\theta_1 \cap \theta_2 \cap \theta_3$, car ce dernier représente l'allégeance suspect. Les sept ensembles[6] d'un cadre de discernement comportant trois singletons, et les cinq allégeances correspondantes possibles qui en découlent sont représentés à la figure 3.3.

[5]Selon la notation introduite dans la section 2.2.5, provenant, entre autres, de la page 43 de [25].
[6]ne considérant pas les unions

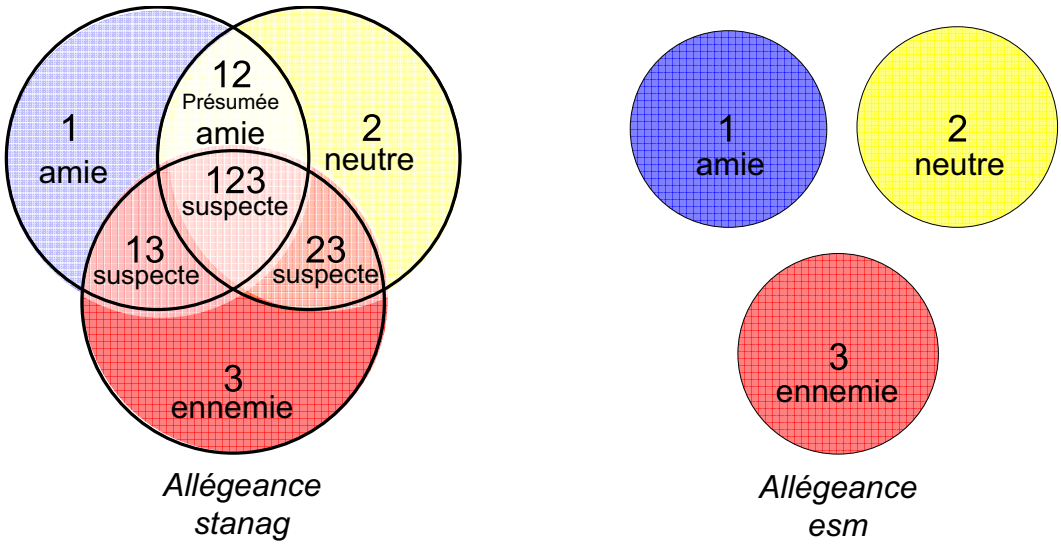


FIG. 3.3 – Représentation sous diagramme de Venn de l'*allégeance stanag* et de l'*allégeance esm* pour un cas sans contrainte

3.1.4 Génération d'informations

Les informations du problème traité dans le cadre de ce mémoire correspondent aux données provenant de capteurs ESM. Pour générer ces informations, nous avons opté pour un système paramétrable permettant de choisir le comportement du capteur ESM simulé. À moins d'avis contraire, nos analyses, comparaisons et simulations procèdent comme suit :

- le capteur ESM détecte l'*allégeance esm* de la cible θ_{ex} avec une probabilité de détection ρ,
- les autres singletons de Θ se partagent le reste de la probabilité $1-\rho$ également,
- pour chaque rapport d'évènement affirmant une identité θ_o, les cibles désignées comme appartenant à l'identité θ_o se voient attribuer une masse μ,
- la masse résiduelle, $1-\mu$, est attribuée à l'ignorance totale I_t,
- durant la première moitié d'une simulation de n itérations, l'identité présumée de la cible est $\theta_{ex} = \theta_\alpha$,
- la seconde moitié de la simulation, un changement d'allégeance fait que la cible se retrouve maintenant avec une identité $\theta_{ex} = \theta_\beta$.

Bien entendu, toutes les variables mentionnées ci-dessus se trouvent à être des paramètres de l'implantation logicielle du simulateur. Elles ont été choisies ainsi afin d'observer les divers comportements d'un système de fusion, dont celui face à des relevés erronés, ou celui face à un changement d'allégeance.

3.1.5 Contraintes appliquées

Les contraintes sont également fixées afin que le cadre de raisonnement D^Θ corresponde à celui d'un pays en état de paix comme sur la figure 3.1.1, à moins d'avis contraire. Dans le cas des règles définies dans l'ensemble de puissance 2^Θ, les contraintes seront ajoutées de façon à réduire l'ensemble d'hyperpuissance à un ensemble de puissance.

3.1.6 Combinaison d'informations avec ou sans filtre

Trois types de filtres ont été implantés pour éviter que le système de fusion ne réponde trop promptement lors de relevés erronés. En premier lieu, il y a un filtre qui laisse tout passer ; autrement dit, aucun filtrage. Il y a un filtre appliqué à la toute fin directement sur les résultats de la combinaison. Finalement, nous avons eu l'idée d'appliquer un filtre sur les entrées de la combinaison. Ce type de filtre équivaut en fait à atténuer l'impact des données à fort potentiel conflictuel lors des combinaisons, évitant ainsi d'induire en erreur[7] le système.

3.1.7 Fonction de masse versus Probabilité pignistique

À priori, les décisions après combinaisons pourraient être prise à partir de la fonction de masse[8] ou à partir de la fonction de probabilité pignistique obtenue de la fonction de masse. La probabilité pignistique correspond à la transformation pignistique généralisée sous D^Θ, et à la transformation pignistique classique sous 2^Θ. Il est tout de même possible de procéder par la transformation pignistique généralisée sous D^Θ et d'appliquer les contraintes sur toutes les intersections, ce qui fait en sorte que l'on agisse sous 2^Θ. On retrouve alors l'équivalent de la transformation pignistique classique sous 2^Θ.

[7] ou mettre dans l'incertitude

[8] En pratique, la décision à partir de la fonction de masse ne se fait jamais, et nos expérimentations nous montre également une incapacité à prendre des décisions ainsi.

3.2 Critères de performance

Dans toute analyse comparative, non seulement la comparaison doit se faire sous les mêmes conditions, mais la classification doit se faire, autant que possible, avec des critères de performance bien établis et invariants pour chacun des éléments à l'étude. Cette section présente les critères privilégiés.

3.2.1 Résistance aux fluctuations dans les relevés ESM

La résistance aux fluctuations dans les relevés ESM peut être vue comme la capacité à conserver son opinion lors d'une combinaison avec des relevés ESM erronés de façon singulière[9] ou en salve[10].

3.2.2 Réactance aux transferts d'allégeance

La réactance aux transferts d'allégeance représente la capacité d'adapter la décision lors d'un changement d'allégeance de la cible. Le système doit être en mesure de rapidement distinguer un changement d'allégeance d'une cible par rapport à une erreur dans un relevé ESM.

3.2.3 Réactance versus Résistance

Comme on le constate, les critères de résistance aux fluctuations ESM et de réactance aux transferts d'allégeance sont plutôt contradictoires. La clef consiste a atteindre une balance acceptable entre les deux critères, selon le contexte. C'est une décision délicate dont l'objectif dépend grandement de l'utilisation finale.

[9] une seule erreur de temps à autre

[10] en séquence de plusieurs erreurs groupées

3.2.4 Validité de décision par la fonction de masse

La validité de décision par la fonction de masse correspond à la capacité à prendre de bonnes décisions à partir du maximum de la fonction de masse. Puisque le contrôle des données réelles et des données fournies est total, il est possible de connaître la vérité et donc de savoir ce qu'un capteur idéal devrait signaler. C'est par rapport à ce capteur idéal que l'on établit cette validité de décision.

Notez qu'en pratique il ne se prend pas de décision à partir de la fonction de masse. Les résultats peu intéressants obtenus lors de l'exploration de cette voie nous en ont dissuadés également. La préférence étant ainsi allée à la décision par la probabilité pignistique.

Taux de bonnes décisions

Par taux de bonnes décisions on entend, le taux de reconnaissance de l'allégeance réelle de la cible simulée. Il est déterminé par un ratio du nombre de fois où le maximum de la fonction de masse, ou pignistique le cas échéant, nous donne le singleton faisant référence à l'allégeance réelle de la cible. La section 3.2.6 en dit plus sur le sujet.

3.2.5 Validité de décision par la probabilité pignistique

La validité de décision par la probabilité pignistique ressemble à celle précédente, mais à partir du maximum de la fonction de distribution de probabilité pignistique. Comme la précédente d'ailleurs, on procède en comparaison avec la vérité obtenue par un capteur idéal.

3.2.6 Taux de succès de la décision

Le taux de succès mesure le ratio du nombre de bonnes décisions prises dans une série de simulations du système de fusion avec les conditions de simulation prédéfinies. Ceci permet d'établir la qualité d'une règle de combinaisons dans ces conditions choisies. Dans le cadre de ce mémoire, on définit la bonne décision comme étant l'identification avec succès du singleton représentant la cible θ_{ex}. Donc, l'ensemble ou singleton qui aura la masse ou probabilité supérieure sera l'ensemble auquel on présumera appartenir

la cible. La décision sera ainsi considérée bonne si l'ensemble de masse ou probabilité supérieure est le singleton qui représente la cible en cours d'analyse. Il faut toutefois savoir qu'un succès, ou bonne décision, se définit autrement sur le terrain selon le contexte. Par exemple, lorsqu'une cible est identifiée comme étant ami, présumé ami, ou neutre, elle ne sera pas l'objet d'attaque ; la décision dans ce cas sera donc de ne pas procéder agressivement, et ce sera le bon choix. Ainsi, selon différents contextes, différentes décisions peuvent être considérées bonnes ou mauvaises.

3.3 Conception, implantation et simulations

3.3.1 Système de simulation

Une grande partie du système de simulation qui touche plus spécifiquement la combinaison d'informations sous D^{Θ}, a déjà fait l'objet d'une publication de l'auteur. La consultation de cette publication [6] fournit plus de détails. Cette section ne présente donc qu'un bref aperçu du système. La figure 3.4 donne une idée générale de l'organisation du système de simulation. On y retrouve différentes règles de combinaisons, dont la règle EACR définie à la page 73.

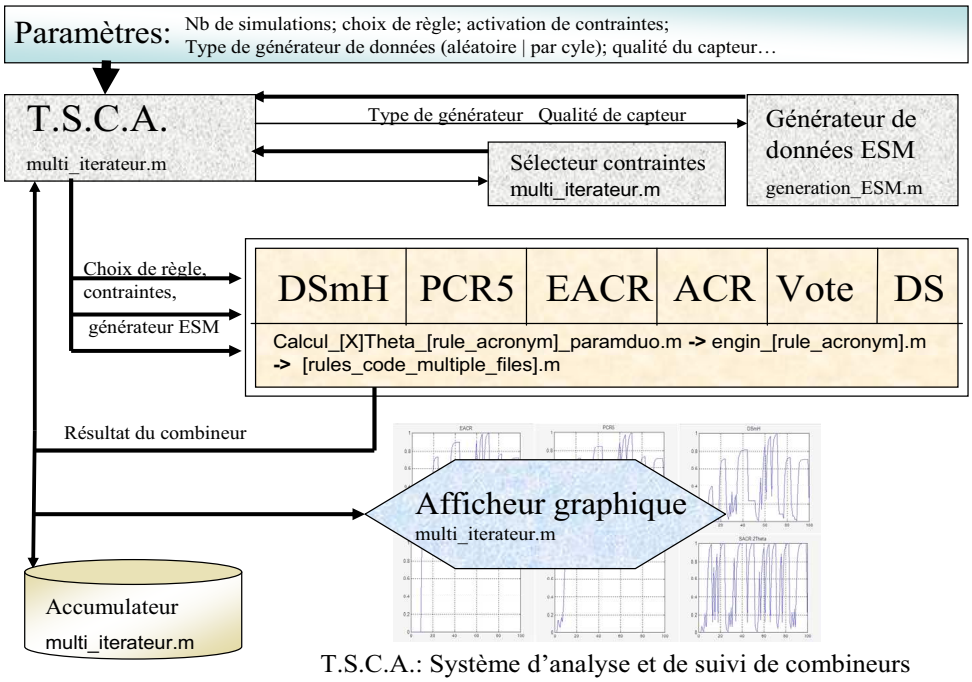


FIG. 3.4 – Organisation du système de simulation

Caractéristiques générales

Notre système, implanté sous Matlab[11], comporte des avantages et désavantages, comme tout logiciel. Matlab est implémenté pour optimiser le calcul matriciel, les données sont conservées dans ce format et les opérations se réalisent principalement sur des matrices. Aussi, notre système est conçu de façon modulaire. Cette modularité est un avantage lorsque l'on souhaite une réutilisation de certaines parties pour d'autres usages que ce pour quoi elles ont été originellement écrites. Par contre, une grande modularité du système peut devenir un désavantage lors du déverminage. Si le message d'erreur est clair, il devient plus aisé de localiser l'erreur en question, autrement le déverminage devient bien plus ardu avec la modularité.

Représentation de l'information

La représentation de l'information a été choisie avec attention. En effet, selon le choix, la complexité peut rapidement s'accroître jusqu'à devenir un problème important. Sans optimisation à ce niveau il faudrait conserver toute l'information ou tous les ensembles possibles sous D^Θ, conserver (ou évaluer à chaque fois) toutes les possibilités d'intersection et d'union pour toutes les possibilités d'ensembles. Il faudrait également maintenir toute cette information en mémoire. Il faudrait également normer la façon de saisir et présenter l'information, implanter un interpréteur en conséquence dans Matlab et qu'il puisse opérer sur ce format choisi. Autrement une implantation d'un module de traduction vers (et en provenance de) un format sur lequel Matlab peut opérer. Dans [25], on y présente au chapitre 2, la partie chargée de générer l'ensemble D^Θ, ce module emploie une méthode d'une complexité s'accroissant à un rythme au-delà de l'exponentiel. Dans le cas de [25], au chapitre 3, le choix de représentation de l'information s'était arrêté sur une table binaire indiquant la présence d'un ensemble parmi les ensembles possibles.

Dans notre cas, nous avons choisi une représentation de l'information par sommes de produits[12] (SOP). Cela aura eu pour effet de grandement simplifier la représentation de l'information. Cette SOP sera implantée vectoriellement au sein de matrices où les vecteurs à l'intérieur de cellule matricielle pourront représenter les produits, et où les rangées/colonnes de cellules composent les sommes.

[11] Matlab est une marque de commerce enregistrée par Mathworks incorporated.

[12] c.-à-d. des unions d'objets de D^Θ qui sont formés d'intersections de singletons.

Somme de produits (SOP)

Le système que nous avons conçu traite l'information en termes d'union d'intersections ou somme de produits. La somme (ADD) étant représentée par l'union ($\cup$), et le produit (MULT) par l'intersection ($\cap$). Nous avons choisi cela, au lieu du produit de sommes (POS) afin d'éviter d'avoir à traiter les parenthèses. On pourrait également utiliser les principes développés pour les circuits logiques par les tables de Karnaugh, les règles booléennes, etc. Voici quelques exemples de notations :

- $\theta_1 \cap \theta_2 \cap \theta_3 = \theta_1\theta_2\theta_3 = [1, MULT, 2, MULT, 3]$
- $\theta_1 \cup \theta_2 \cup \theta_3 = \theta_1 + \theta_2 + \theta_3 = [1, ADD, 2, ADD, 3]$
- $(\theta_1 \cap \theta_2) \cup \theta_3 = \theta_1\theta_2 + \theta_3 = [1, MULT, 2, ADD, 3] = [3, ADD, 1, MULT, 2]$
- $(\theta_1 \cup \theta_2) \cap \theta_3 = (\theta_1 \cap \theta_3) \cup (\theta_2 \cap \theta_3) = \theta_1\theta_3 + \theta_2\theta_3$
  $= [1, MULT, 3, ADD, 2, MULT, 3]$
- $(\theta_1 \cap \theta_2 \cap \theta_3) \cup (\theta_4 \cap \theta_5) = \theta_1\theta_2\theta_3 + \theta_4\theta_5$
  $= [1, MULT, 2, MULT, 3, ADD, 4, MULT, 5]$

Conversion entre les notations *somme de produits* et *produit de sommes*

Tel que mentionné, nous utilisons la somme de produits comme principale méthode pour noter les ensembles. Toutefois, comme on le verra, nous utilisons le produit de sommes ou intersections d'unions dans quelques parties du système de combinaison afin de simplifier le processus de calcul. Plus spécifiquement, ce double système de notation introduit dans les deux dernières colonnes de la table 3.1, est fait de façon à ce que l'on puisse utiliser le même algorithme pour fonctionner avec la matrice des unions[13] ainsi que la matrice des intersections[14]. La table 3.1 présente ainsi la notation utilisée dans la suite, accompagnée de son équivalent en notation mathématique. Nous voyons dans la notation SOP de la table 3.1, une ligne représentant un monôme d'un produit (e.g. $\theta_1\theta_3$) additionnée à un autre monôme θ_2 pour obtenir l'union (e.g. $\theta_1\theta_3 + \theta_2$). Dans la notation par POS, nous montrons une situation où une ligne représente un monôme d'une somme (e.g. $\theta_1 + \theta_3$) multipliée à un autre monôme θ_2 pour obtenir l'intersection (e.g. $\theta_2\,(\theta_1 + \theta_3)$).

L'étape difficile est la conversion d'un ensemble sous la notation de SOP en celle de POS. Pour de simples cas, tels que ceux présentés dans les trois premières lignes

[13]Matrice résultant d'une combinaison par union de deux corps d'évidence qui comporte toutes les unions ayant lieu lors de cette combinaison.

[14]Matrice résultant d'une combinaison par intersection de deux corps d'évidence qui comporte toutes les intersections ayant lieu lors de cette combinaison.

TAB. 3.1 – Notations équivalentes pour des évènements

Mathématiques	Entrées/Sorties Matlab	SOP	POS
$\{\theta_1\}$	$[1]$	$\begin{bmatrix} 1 \end{bmatrix}$	$\begin{bmatrix} 1 \end{bmatrix}$
$\{\theta_1 \cup \theta_2\}$	$[1, ADD, 2]$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \end{bmatrix}$
$\{\theta_1 \cap \theta_2\}$	$[1, MULT, 2]$	$\begin{bmatrix} 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$
$\{(\theta_2) \cup (\theta_1 \cap \theta_3)\}$	$[2, ADD, 1, MULT, 3]$	$\begin{bmatrix} 2 & \\ 1 & 3 \end{bmatrix}$	–
$\{(\theta_1 \cup \theta_2) \cap (\theta_2 \cup \theta_3)\}$	–	–	$\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$
$\{(\theta_1 \cap \theta_2) \cup (\theta_2 \cap \theta_3)\}$	$[1, MULT, 2, ADD, 2, MULT, 3]$	$\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$	–
$\{(\theta_2) \cap (\theta_1 \cup \theta_3)\}$	–	–	$\begin{bmatrix} 2 & \\ 1 & 3 \end{bmatrix}$

de la table 3.1, cela consiste seulement à changer les lignes des matrices en colonnes et les colonnes en lignes. Aux fins de simplifications dans le processus de conversion, nous utilisons la règle d'absorption $\theta_1 + \theta_1\theta_2 = \theta_1$ qui est dérivée du fait que $(\theta_1\theta_2) \subseteq \theta_1$. En utilisant cette règle, on peut voir comment on en vient aux deux dernières rangées de la table 3.1. Le processus est détaillé par les équations (3.1) et (3.2).

$$\begin{aligned} (\theta_1 \cup \theta_2) \cap (\theta_2 \cup \theta_3) &= (\theta_1 + \theta_2)(\theta_2 + \theta_3) \\ &= \theta_1\theta_2 + \theta_1\theta_3 + \theta_2 + \theta_2\theta_3 \\ &= \theta_1\theta_3 + \theta_2 \end{aligned} \tag{3.1}$$

$$\begin{aligned} (\theta_1 \cap \theta_2) \cup (\theta_2 \cap \theta_3) &= \theta_1\theta_2 + \theta_2\theta_3 \\ &= \theta_2(\theta_1 + \theta_3) \end{aligned} \tag{3.2}$$

Toutefois, dans le code Matlab programmé, les procédures suivantes sont utilisées et fonctionnent pour tous cas. Elles se basent sur l'utilisation de la loi de DeMorgan telle que présentée par les équations (3.3) et (3.4). En procédant à la transformation de

la loi de DeMorgan deux fois, on peut éviter l'utilisation d'ensembles complémentaires. Ainsi, on respecte la théorie DSm malgré l'utilisation de cette loi mathématique qui implique des compléments. L'utilisation de la règle d'absorption aide également pour obtenir une meilleure simplification du calcul.

$$\overline{A}\,\overline{B} = \overline{A \,+\, B} \tag{3.3}$$

$$\overline{A} \,+\, \overline{B} = \overline{A\,B} \tag{3.4}$$

Voici comment on procède pour le cas de la conversion d'un ensemble en notation SOP à une notation POS. C'est plutôt simple en fait, on débute par l'inversion des opérateurs (en changeant les additions ($\cup$) pour des multiplications ($\cap$) et vice versa), suivis d'une distribution des produits et d'une étape de simplification. On conclut avec une seconde inversion des opérateurs. Puisque l'on aura utilisé l'inversion deux fois, on n'obtiendra pas l'opérateur de négation, ($\overline{\overline{A}} = A$).

Exemple 3.1 Conversion de notation SOP à POS

Supposons que l'on veut convertir l'ensemble montré à l'équation (3.5) à un ensemble en notation en produit de sommes. Une simple distribution des produits suivis d'une simplification permettront de parcourir le chemin inverse.

$$\begin{aligned}
\theta_1 + \theta_2\theta_3 + \theta_2\theta_4 &= \overline{\overline{(\theta_1)}\,\left(\overline{\theta_2} + \overline{\theta_3}\right)\,\left(\overline{\theta_2} + \overline{\theta_4}\right)} \\
&= \overline{\overline{\theta_1}\,\overline{\theta_2} + \overline{\theta_1}\,\overline{\theta_2}\,\overline{\theta_4} + \overline{\theta_1}\,\overline{\theta_2}\,\overline{\theta_3} + \overline{\theta_1}\,\overline{\theta_3}\,\overline{\theta_4}} \\
&= \overline{\overline{\theta_1}\,\overline{\theta_2} + \overline{\theta_1}\,\overline{\theta_3}\,\overline{\theta_4}} \\
&= (\theta_1 + \theta_2)\,(\theta_1 + \theta_3 + \theta_4)
\end{aligned} \tag{3.5}$$

Générateur de relevé ESM

Notre générateur de relevé ESM modulaire offre la possibilité de manipuler un grand nombre de paramètres. Il permet de choisir le taux d'occurrence d'un singleton principal ainsi que la masse qui y est attribuée. Il est possible également de demander au système de procéder à un changement d'allégeance après un certain pourcentage du nombre

d'itérations à exécuter. Enfin, il est aisé pour un utilisateur de modifier le code pour permettre de nombreuses variantes ou différents paramétrages.

Principes derrière la conjonction et la disjonction

Tel qu'expliqué dans [6], nous avons choisi la représentation sous la notation en somme de produits (SOP). Ce choix a permis, entre autres, d'éviter la gestion de parenthèses et des priorités qu'aurait nécessité une autre représentation. Nous avons également implémenté des modules qui effectuent la transition entre la somme de produits (SOP) et le produit de sommes (POS). Nous avons aussi implémenté un module[15] qui procède à une multiplication des monômes un à un dans le module de conjonction[16] ainsi que dans le module de disjonction[17]. Ces deux modules sont identiques à deux points près, dans le cas de la conjonction, le module `intersection_matrice.m` utilise la matrice des monômes en POS, et après l'exécution de ce module, il y a appel du convertisseur en notation SOP. C'est donc exactement la même procédure qui s'applique dans nos modules de calculs de conjonction et de disjonction, hormis le fait que chacun s'exécute sur une matrice de monômes dans une notation différente.

Traitement des conflits

Notre système est en mesure de faire un traitement dynamique ou statique des contraintes. En dynamique, il tient compte des contraintes à chaque étape des combinaisons et même d'avoir des contraintes qui diffèrent d'une étape à l'autre. En statique, les contraintes sont fixes et appliquées à la fin du traitement en bloc. Une validation de la présence de contraintes au sein du code des combinaisons est employée afin de procéder selon les conditions des règles de combinaisons.

Un module/fonction permet la vérification d'une ou plusieurs informations sur lesquelles s'appliquent les contraintes et sort les informations exemptes des ensembles sur lesquelles sont appliquées des contraintes. Ce module fait également appel au module de fusion des doublons que l'on retrouve à la fin d'une combinaison. (e.g. θ_1 apparaît 3 fois dans le tableau 2.2.)

[15] intersection_matrice.m
[16] engin_conjonction.m
[17] engin_disjonction.m

3.3.2 Règles de combinaison

Cette section effectue un tour d'horizon de chacune des règles de combinaison implantée et relève les principaux points et particularités. Le noyau du code développé se retrouve aussi présenté dans [6]. Tout le reste du système a été élaboré à partir de ce noyau. Puisque la DSmH est la plus complexe au niveau algorithmique et que son développement est déjà fait et présenté dans [6], il n'est pas nécessaire de passer en long et en large chacune des règles à l'étude, toutes plus simples que la DSmH. Leur développement ne consiste qu'à simplifier, ou réduire le code de la DSmH.

Vote majoritaire

Le module nécessaire au calcul de la combinaison par vote majoritaire se retrouve dans le fichier `engin_vote.m`. Ce dernier est simplement constitué d'une série de boucles permettant les décomptes nécessaires au calcul de moyennes et de ratios afin de procéder au vote majoritaire tel que présenté à la section 2.2.6.

Conjonction

Le module nécessaire à la règle de combinaison conjonctive se retrouve dans le fichier `engin_conjonction.m` et est déjà développé dans [6] puisque partie intégrante du calcul de la DSmH.

Disjonction

Le module nécessaire à la règle de combinaison disjonctive se retrouve dans le fichier `engin_disjonction.m` et est déjà développé dans [6] puisque partie intégrante du calcul de la DSmH.

Dempster (DS)

Également basé sur le code déjà développé pour la DSmH, le module de calcul pour la règle de Dempster apparaît dans le fichier `engin_dst.m`. Il applique la redistribution à tous les éléments focaux non conflictuels par le biais d'un facteur de normalisation

basé sur la valeur du conflit conjonctif. Il s'agit donc d'une simple multiplication de la matrice de l'intersection modifiée par le retrait des masses du conflit. L'exemple 2.5 du chapitre 2 porte sur l'utilisation de la règle de DS de façon théorique. Voici maintenant un exemple considérant les mêmes données mais réalisé au sein de Matlab.

Exemple 3.2 Exemple (2.5) réalisé dans notre système sous Matlab

Voici d'abord les données à combiner de l'exemple (2.5) tel que représentées dans Matlab[18].

```
>> info(1)
ans =
    elements: {[1]  [2]  [1 -2 2 -2 3]}
      masses: [0.6000 0.2000 0.2000]
>> info(2)
ans =
    elements: {[1]  [2]  [1 -2 2 -2 3]}
      masses: [0.2000 0.2000 0.6000]
>> info(3)
ans =
    elements: {[1]  [3]  [1 -2 2 -2 3]}
      masses: [0.5000 0.4000 0.1000]
```

Voici maintenant les résultats de combinaison DS à l'étape intermédiaire, puis suite à la seconde combinaison. La combinaison a bien évidemment eu lieu de façon dynamique, autrement il n'y aurait eu qu'une seule étape. Aussi, on a appliqué les contraintes de l'exemple à chaque étape de combinaison.

```
DST  % etape intermediaire
1 : m=0.61904762
2 : m=0.23809524
1 ADD  2 ADD  3 : m=0.14285714

DST  % resultat final
1 : m=0.82300885
```

[18] [1 -2 2 -2 3] représente [1 ADD 2 ADD 3] donc $1 \cup 2 \cup 3$. Ainsi, -2 représente l'union, et -1 l'intersection.

```
3 : m=0.10619469
2 : m=0.04424779
1 ADD 2 ADD 3 : m=0.02654867
```

On constate que les résultats de la combinaison sont identiques à ceux de l'exemple (2.5).

Règle de combinaison adaptative (ACR)

La règle de combinaison adaptative procède presque comme celle de Dempster. Elle est également basée sur le code déjà développé pour la DSmH. Le module de calcul pour la règle ACR, retrouvé dans le fichier `engin_acr.m`, se comporte tel que prévu par la théorie sans complication ou particularité algorithmique.

ACR-étendu (EACR, Extended-ACR)

Une section entière est dédiée à la règle de combinaison dans le prochain chapitre. Proche parent des règles ACR et DSmH, le code développé pour la EACR est également issu de celui de la DSmH.

Règle de combinaison de Dezert-Smarandache Hybride (DSmH)

Le code de base de DSmH est présenté de façon très détaillée dans [6], également en annexe. Il serait bon de consulter cette référence avant de poursuivre, car elle est la règle la plus complexe au niveau algorithmique. Développée avant les autres, elle sert de base aux autres règles. Elle comporte un grand nombre de modules implantés dont les autres règles de combinaisons utilisent intégralement à quelques modifications près.

Le module principal de cette règle, `engin_dsmh.m` se subdivise en trois parties ; l'initialisation des variables et la création des matrices d'intersections et d'unions, l'exécution de la procédure décrite par la table 2.15, et puis enfin il y a le post-traitement. Cette dernière étape consiste en une étape de tri des données, de fusion des doublons, et d'application d'une masse minimale donnée à l'ignorance totale.

L'exécution de la procédure décrite par la table 2.15 dans le code Matlab du module `engin_dsmh.m` se déroule dans des boucles qui valident la présence de contraintes et

qui, selon l'étape[19] où est rendu le calcul, applique différentes combinaisons des matrices d'unions et d'intersections créées à la partie d'initialisation du module `engin_dsmh.m`.

Proportional Conflict Redistribution Rule 5 (PCR5)

La complexité algorithmique de cette règle de combinaison demeurant inférieure à la DSmH, et considérant que la quasi-totalité des modules requis pour cette règle est déjà implantée, nous avons encore une fois repris le code de la DSmH pour le développement de cette règle-ci. L'engin de calcul de la PCR5 se retrouve dans le fichier `engin_pcr.m` et permet le calcul de la combinaison suivant la théorie.

3.3.3 Quasi-associativité

Exception faite des combinaisons Dempster, conjonctive et disjonctive, les autres règles de combinaison ne sont pas associatives ; l'ordre de combinaison est donc important.

La quasi-associativité est rendue possible en traitant toutes les informations en un seul bloc et non de façon séquentielle. On peut aussi exploiter l'associativité de règles de base (conjonctive et disjonctive) et distribuer la masse du conflit à la toute fin de la combinaison ce qui revient à en faire un traitement en bloc.

Pour faire les calculs des règles de combinaisons avec la quasi-associativité présentée dans la section 4.3.1, un module principal du système est cloné et est modifié à l'endroit où sont appelées les règles de combinaisons. Il procède selon la théorie montrée à la section 4.3.1.

Comme le montre la figure 3.5, la règle conjonctive est exécutée en boucle utilisant l'information de la règle conjonctive précédente. Au moment de prendre une mesure au temps τ_n, le système procède à la redistribution de la masse du conflit selon la règle choisie avec les données sorties de la conjonction du temps τ_{n-1}. La quasi-associativité modifiée pour fonctionnement sous D^Θ, exige une seconde boucle de calcul qui est exécutée en parallèle avec la règle disjonctive. Le module `compute_combine_qa.m` appelle les modules des règles de combinaison selon la procédure de quasi-associativité.

[19] On réfère ici aux étapes de la procédure de la table 2.15.

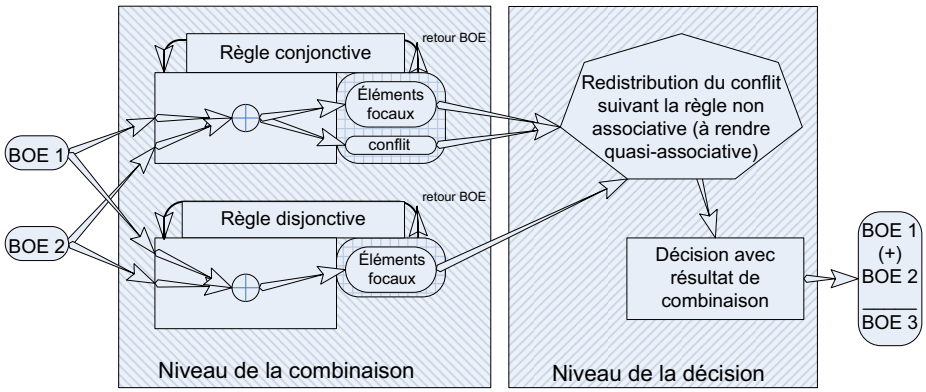


FIG. 3.5 – Schéma de l'algorithme de quasi-associativité

3.3.4 Transformation Pignistique Généralisée

Cette section effectue un tour de chacun des modules impliqués dans la transformation pignistique généralisée (TPG) (voir page 37) qui est implantée ; elle relève aussi les principaux points et particularités.

Cardinal sous l'ensemble d'hyperpuissance

Le cardinal DSm calculé ici est tel que la théorie le prescrit dans [7]. On procède d'abord par la création de l'ensemble d'hyperpuissance puis par l'élimination des éléments qui sont inclus dans les contraintes. Le décompte du nombre d'éléments restant donne le cardinal DSm. Dans cette procédure, on désigne par le terme *dimension*, le cardinal d'un ensemble, qui peut être l'ensemble de puissance, ou même l'ensemble d'hyperpuissance. La fonction `calcul_cardinal.m` procède comme suit :

1. Saisie des contraintes en paramètre de la fonction[20]
2. Génération de deux ensembles de puissance (2^Θ), soit 2^Θ_α et 2^Θ_β dont le nombre d'objets est mise dans une variable temporaire n_ϵ.
3. Pour chacun des n_ϵ ensembles de l'ensemble de puissance 2^Θ_α, on utilise la fonction `test_contrainte.m` pour valider si l'ensemble est contraint. Afin de voir si une contrainte doit s'appliquer sur cet ensemble de puissance.
   (a) Si l'ensemble est contraint, le cardinal DSm de cet ensemble sera de 0.
4. Génération d'un ensemble d'hyperpuissance D^Θ

[20] L'exemple 2.12 démontrait l'influence des contraintes sur le Cardinal DSm.

(a) On génère un ensemble d'hyperpuissance (D^Θ) désigné D_ρ^Θ et on initialise les valeurs de cardinaux DSm à 0.

(b) On note le nombre d'objets de l'ensemble d'hyperpuissance D_ρ^Θ : n_ρ

(c) On sépare en objets élémentaires (c.-à-d. monômes) les ensembles de l'ensemble d'hyperpuissance à l'aide de la fonction `separation.m`.

5. Pour chacun des n_ρ ensembles de l'ensemble d'hyperpuissance D_ρ^Θ, soit pour j allant de 1 à n_ρ :

(a) Remise à zéro d'un compteur principal du cardinal DSm.

(b) Pour chacun des n_ϵ ensembles de l'ensemble de puissance 2_β^Θ, soit pour k allant de 1 à n_ϵ :

i. Remise à zéro d'un compteur secondaire du cardinal DSm.

ii. On met l'ensemble k de 2_β^Θ dans une variable temporaire $T1$

iii. On note la longeur de l'élément j de l'ensemble d'hyperpuissance D_ρ^Θ séparé en objets élémentaires : $n_{\rho s}$

iv. Pour chacun des $n_{\rho s}$ éléments de l'ensemble d'hyperpuissance D_ρ^Θ séparé en objets élémentaires, soit pour l allant de 1 à $n_{\rho s}$:

A. On lui compare $T1$, s'il y a équivalence, on met 1 dans le compteur secondaire du cardinal DSm.

v. On incrémente le compteur principal du cardinal DSm avec la valeur du compteur secondaire.

(c) On place enfin la valeur du compteur principal du cardinal DSm dans l'espace mémoire qui lui est réservé pour l'ensemble en cours.

Transformation pignistique généralisée

Suite à la création d'une table des cardinaux DSm du cas présent, nous effectuons simplement l'algorithme développé pour l'exécution de l'équation de la transformation pignistique généralisée. Voir l'équation 2.35 à la section 2.5.4.

3.3.5 Complexité de la génération de l'ensemble d'hyperpuissance

Cette section aborde le problème de la complexité survenant lors de la génération d'ensemble d'hyperpuissance. Ce problème est directement lié au grand besoin en puissance de calculs, mais surtout en espace mémoire nécessaire aux calculs. Le problème

incontournable est que l'on travaille avec des ensembles de dimensions s'accroissant à un rythme phénoménal tel que mentionné à la section 2.4.1.

Afin de mieux saisir le problème, on présente le rythme de croissance de la séquence de Dedekind de la table 2.14 en comparaison avec d'autres séquences dans la table 3.2,

TAB. 3.2 – Différentes séries de nombres

n	1	2	3	4	5	6	7	8
Fibonacci	1	1	2	3	5	8	13	21
Premiers	3	5	7	11	13	17	19	23
2^n	2	4	8	16	32	64	128	256
3^n	3	9	27	81	243	729	2187	6561
$\lfloor n * exp(n) \rfloor$	2	14	60	218	742	2420	7676	23847
10^n	10	100	1000	10000	100000	1×10^6	1×10^7	1×10^8
Factoriel ($n!$)	1	2	6	24	120	720	5040	40320
Superfactoriel	1	2	12	288	34560	24×10^6	12×10^{10}	5×10^{15}
Dedekind	2	5	19	167	7580	78×10^5	24×10^{11}	56×10^{21}

Notez que le nombre de Dedekind a un impact sur nos calculs dans le sens où il indique le nombre d'ensembles maximum possibles pour un cadre de raisonnement DSm. En fait, c'est le nombre d'ensembles contenus dans l'ensemble d'hyperpuissance pour un cardinal du cadre de discernement donné. Ainsi, pour un cardinal de cadre de discernement de 5, l'ensemble d'hyperpuissance correspondant a 7580 ensembles. Ces derniers vont de l'ensemble vide $\emptyset$, à l'ensemble union totale $\theta_1 \cup \theta_2 \cup \theta_3 \cup \theta_4 \cup \theta_5$, en passant par des ensembles comme $(\theta_1 \cap \theta_5) \cup (\theta_2 \cap \theta_3) \cup (\theta_1 \cap \theta_2 \cap \theta_4 \cap \theta_5)$. Suivant la méthode de calcul, il peut être requis de maintenir en mémoire plusieurs variables contenant tous les ensembles, mais il faut en plus les exprimer car certains comprennent des combinaisons complexes d'unions et d'intersections. Notre méthode, telle que présentée dans [6], exige lors d'intersection par exemple, de maintenir au moins deux matrices ayant k^2 cellules, avant la description de l'ensemble ainsi que sa masse. De plus, de multiples boucles `for` sont requises pour le filtrage de doublons, la mise en ordre de grandeur des ensembles (c.-à-d. $1 \cap 3$ vient avant $1 \cap 2 \cap 3$), la mise en ordre numérique des ensembles (c.-à-d. 1 vient avant 2) et les autres tâches.

Nous avons été en mesure de générer les ensembles d'hyperpuissance pour différents cas. On a également noté[21] le temps et l'espace mémoire requis.

[21] Ces mesures ont été notées à partir d'un portable ayant un processeur Céléron mobile 1.6 GHz et une mémoire vive de 2 Gb.

Ensemble d'hyperpuissance pour le cas $|\Theta| = 3$

L'expression d'un ensemble d'hyperpuissance se réalise par le biais de plusieurs fonction. Celle qui a exigé le plus de mémoire en demandait approximativement 160 Ko. L'exécution a pris 1.4 secondes, prenant le temps d'afficher toutes les informations sur les étapes du calcul en cours.

```
 DISPLAY HYPERPOWER SET, of size 18

1    1 ADD 2    1 MULT 2
2    1 ADD 3    1 MULT 3
3    2 ADD 3    2 MULT 3

1 MULT 2 MULT 3
2 ADD 1 MULT 3
1 ADD 2 MULT 3
3 ADD 1 MULT 2
1 ADD 2 ADD 3

1 MULT 3 ADD 2 MULT 3
1 MULT 2 ADD 2 MULT 3
1 MULT 2 ADD 1 MULT 3
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3
```

Ensemble d'hyperpuissance pour le cas $|\Theta| = 4$

Dans ce cas-ci, il a fallu près de 33.7 secondes, ainsi que plus de 3 Mo d'espace mémoire.

```
 DISPLAY HYPERPOWER SET, of size 166

1           1 MULT 4          2 ADD 1 MULT 4   3 ADD 1 MULT 2
2           2 ADD 3           1 ADD 2 ADD 3    3 ADD 1 MULT 4
3           2 ADD 4           1 ADD 2 ADD 4    1 ADD 2 MULT 3
4           2 MULT 4          1 ADD 3 ADD 4    1 ADD 2 MULT 4
1 ADD 2     3 ADD 4           2 ADD 3 ADD 4    3 ADD 2 MULT 4
1 MULT 2    3 MULT 4          2 MULT 3 MULT 4  2 ADD 3 MULT 4
```

```
1 ADD 3     1 MULT 2 MULT 3   4 ADD 1 MULT 2   1 ADD 2 ADD 3 ADD 4
2 MULT 3    1 MULT 3 MULT 4   4 ADD 2 MULT 3   1 MULT 2 MULT 3 MULT 4
1 MULT 3    1 MULT 2 MULT 4   4 ADD 1 MULT 3   2 ADD 1 MULT 3 MULT 4
1 ADD 4     2 ADD 1 MULT 3    1 ADD 3 MULT 4   2 ADD 4 ADD 1 MULT 3
```

```
2 ADD 3 ADD 1 MULT 4     1 MULT 3 ADD 1 MULT 4
3 ADD 1 MULT 2 MULT 4    1 MULT 3 ADD 2 MULT 4
2 MULT 4 ADD 3 MULT 4    1 MULT 4 ADD 2 MULT 4
3 ADD 4 ADD 1 MULT 2     1 MULT 3 ADD 3 MULT 4
2 MULT 3 ADD 2 MULT 4    1 MULT 4 ADD 3 MULT 4
2 MULT 3 ADD 3 MULT 4    1 ADD 2 MULT 3 MULT 4
1 ADD 4 ADD 2 MULT 3     1 MULT 2 ADD 1 MULT 3
2 MULT 3 ADD 1 MULT 4    1 MULT 2 ADD 2 MULT 3
1 ADD 3 ADD 2 MULT 4     1 MULT 2 ADD 1 MULT 4
1 MULT 3 ADD 2 MULT 3    1 MULT 2 ADD 2 MULT 4
```

```
1 ADD 2 ADD 3 MULT 4            2 ADD 1 MULT 3 ADD 3 MULT 4
1 MULT 2 ADD 3 MULT 4           2 MULT 4 ADD 1 MULT 2 MULT 3
4 ADD 1 MULT 2 MULT 3           2 MULT 4 ADD 1 MULT 3 MULT 4
1 MULT 3 ADD 1 MULT 2 MULT 4    2 ADD 1 MULT 3 ADD 1 MULT 4
1 MULT 3 ADD 2 MULT 3 MULT 4    1 MULT 2 ADD 2 MULT 3 MULT 4
4 ADD 1 MULT 2 ADD 1 MULT 3     2 MULT 3 ADD 1 MULT 2 MULT 4
4 ADD 1 MULT 2 ADD 2 MULT 3     1 MULT 2 ADD 1 MULT 3 MULT 4
2 ADD 1 MULT 4 ADD 3 MULT 4     2 MULT 3 ADD 1 MULT 3 MULT 4
3 MULT 4 ADD 1 MULT 2 MULT 3    1 ADD 2 MULT 3 ADD 2 MULT 4
3 MULT 4 ADD 1 MULT 2 MULT 4    1 ADD 2 MULT 4 ADD 3 MULT 4
```

```
3 ADD 1 MULT 4 ADD 2 MULT 4         1 MULT 3 ADD 2 MULT 3 ADD 1 MULT 4
1 MULT 4 ADD 1 MULT 2 MULT 3        1 MULT 2 MULT 4 ADD 1 MULT 3 MULT 4
1 MULT 4 ADD 2 MULT 3 MULT 4        1 MULT 2 MULT 4 ADD 2 MULT 3 MULT 4
3 ADD 1 MULT 2 ADD 1 MULT 4         1 MULT 2 ADD 2 MULT 3 ADD 1 MULT 4
3 ADD 1 MULT 2 ADD 2 MULT 4         1 MULT 2 MULT 3 ADD 1 MULT 2 MULT 4
1 ADD 2 MULT 3 ADD 3 MULT 4         1 MULT 2 MULT 3 ADD 1 MULT 3 MULT 4
4 ADD 1 MULT 3 ADD 2 MULT 3         1 MULT 2 MULT 3 ADD 2 MULT 3 MULT 4
2 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4   1 MULT 2 ADD 1 MULT 3 ADD 1 MULT 4
2 MULT 3 ADD 1 MULT 4 ADD 3 MULT 4   1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3
1 MULT 3 MULT 4 ADD 2 MULT 3 MULT 4  1 MULT 2 ADD 1 MULT 3 ADD 3 MULT 4
```

```
1 MULT 2 ADD 1 MULT 4 ADD 2 MULT 4
1 MULT 2 ADD 1 MULT 4 ADD 3 MULT 4
1 MULT 2 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 3 ADD 2 MULT 3 ADD 3 MULT 4
1 MULT 3 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 4
1 MULT 3 ADD 2 MULT 3 ADD 2 MULT 4
1 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4
1 MULT 3 ADD 1 MULT 4 ADD 3 MULT 4
1 MULT 2 ADD 2 MULT 3 ADD 2 MULT 4

1 MULT 2 ADD 2 MULT 3 ADD 3 MULT 4
2 MULT 3 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 4 ADD 2 MULT 4 ADD 3 MULT 4

2 MULT 4 ADD 3 MULT 4 ADD 1 MULT 2 MULT 3
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3 MULT 4
1 MULT 2 ADD 1 MULT 4 ADD 2 MULT 3 MULT 4
1 MULT 3 ADD 1 MULT 4 ADD 2 MULT 3 MULT 4
1 MULT 4 ADD 3 MULT 4 ADD 1 MULT 2 MULT 3
1 MULT 2 ADD 2 MULT 3 ADD 1 MULT 3 MULT 4
1 MULT 2 ADD 2 MULT 4 ADD 1 MULT 3 MULT 4

2 MULT 3 ADD 2 MULT 4 ADD 1 MULT 3 MULT 4
1 MULT 4 ADD 2 MULT 4 ADD 1 MULT 2 MULT 3
2 ADD 1 MULT 3 ADD 1 MULT 4 ADD 3 MULT 4
1 MULT 3 ADD 2 MULT 3 ADD 1 MULT 2 MULT 4
1 MULT 3 ADD 3 MULT 4 ADD 1 MULT 2 MULT 4
2 MULT 3 ADD 3 MULT 4 ADD 1 MULT 2 MULT 4
4 ADD 1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3
1 ADD 2 MULT 3 ADD 2 MULT 4 ADD 3 MULT 4
3 ADD 1 MULT 2 ADD 1 MULT 4 ADD 2 MULT 4

1 MULT 4 ADD 2 MULT 3 ADD 2 MULT 4 ADD 3 MULT 4

1 MULT 3 ADD 1 MULT 4 ADD 2 MULT 3 ADD 2 MULT 4
1 MULT 2 ADD 1 MULT 4 ADD 2 MULT 3 ADD 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3 ADD 1 MULT 4
1 MULT 2 ADD 2 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4
1 MULT 3 ADD 2 MULT 3 ADD 1 MULT 4 ADD 3 MULT 4
```

```
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3 ADD 2 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 3 ADD 1 MULT 2 MULT 4 ADD 2 MULT 3 MULT 4
1 MULT 2 ADD 1 MULT 3 MULT 4 ADD 2 MULT 3 MULT 4
1 MULT 2 ADD 1 MULT 4 ADD 2 MULT 4 ADD 3 MULT 4

2 MULT 3 ADD 1 MULT 2 MULT 4 ADD 1 MULT 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3 ADD 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 1 MULT 4 ADD 3 MULT 4
1 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 2 ADD 2 MULT 3 ADD 2 MULT 4 ADD 3 MULT 4
3 MULT 4 ADD 1 MULT 2 MULT 3 ADD 1 MULT 2 MULT 4
1 MULT 4 ADD 1 MULT 2 MULT 3 ADD 2 MULT 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4
1 MULT 3 ADD 2 MULT 3 ADD 2 MULT 4 ADD 3 MULT 4
2 MULT 4 ADD 1 MULT 2 MULT 3 ADD 1 MULT 3 MULT 4

1 MULT 2 MULT 3 ADD 1 MULT 2 MULT 4 ADD 2 MULT 3 MULT 4
1 MULT 2 MULT 4 ADD 1 MULT 3 MULT 4 ADD 2 MULT 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 1 MULT 4 ADD 2 MULT 3 MULT 4
1 MULT 2 MULT 3 ADD 1 MULT 2 MULT 4 ADD 1 MULT 3 MULT 4
1 MULT 4 ADD 2 MULT 4 ADD 3 MULT 4 ADD 1 MULT 2 MULT 3
1 MULT 2 MULT 3 ADD 1 MULT 3 MULT 4 ADD 2 MULT 3 MULT 4
1 MULT 3 ADD 2 MULT 3 ADD 3 MULT 4 ADD 1 MULT 2 MULT 4
1 MULT 2 ADD 2 MULT 3 ADD 2 MULT 4 ADD 1 MULT 3 MULT 4

1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3 ADD 1 MULT 4 ADD 3 MULT 4

1 MULT 2 ADD 2 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 3 ADD 2 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 1 MULT 4 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 2 ADD 1 MULT 3 ADD 2 MULT 3 ADD
                          1 MULT 4 ADD 2 MULT 4 ADD 3 MULT 4
1 MULT 2 MULT 3 ADD 1 MULT 2 MULT 4 ADD
                        1 MULT 3 MULT 4 ADD 2 MULT 3 MULT 4
```

3.3.6 Taux de succès de la décision

Le module principal a été conçu et développé sous forme d'une fonction dont le nombre d'itérations est l'un des paramètres. Il est donc facile de produire une simulation de type Monte-Carlo en l'insérant dans une boucle simple qui répète l'appel au module principal autant de fois que nécessaire.

3.3.7 Intervalle de confiance des simulations Monte-Carlo

Chaque simulation comprend un nombre fixe d'itérations. Chaque itération correspond à la fusion d'une nouvelle information. Donc, pour obtenir un intervalle de confiance[22] adéquat, on répète la simulation avec les mêmes paramètres un certain nombre de fois (n essais Monte-Carlo) pour former un ensemble de simulations. Tous les résultats sont sauvegardés. Ainsi, à une itération donnée, on a n valeurs provenant des n répétitions. On procède alors aux calculs décrits par les équations (3.6) à (3.8).

Moyenne ($\overline{X}$)

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.6}$$

Variance ($\hat{\sigma}^2$)

$$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \overline{X}\right)^2 \tag{3.7}$$

Intervalle de Confiance (IC)

$$1.96 \cdot \hat{\sigma} \longmapsto \quad IC \;\; de \;\; 95\% \tag{3.8}$$

[22] On présume ici d'une distribution statistique suivant une loi normale.

3.4 Conclusions

L'implantation efficace de la DSmH peut sembler facile à première vue, mais a impliqué beaucoup de réflexions sur les manières possibles de procéder. Ceci a conduit à des solutions particulières pour arriver enfin à reproduire les règles de combinaison dans le cadre de raisonnement complexe qu'est la DSmT. Les premiers résultats de comparaison utilisant cette implantation sont publiés dans [7] (voir aussi en annexe) où on découvre plusieurs avantages et désavantages qui seront présentés au chapitre 5. Lors du développement du code, des idées ont surgi dont certaines méritent de s'y arrêter plus longuement. Les problèmes théoriques rencontrés ainsi que les solutions envisagées et proposées font l'objet du prochain chapitre.

Chapitre 4

Problèmes rencontrés et solutions proposées

Le 'trouble' est une de ces rares choses que l'on trouvera toujours lorsqu'elle est recherchée. – Pascal Djiknavorian

4.1 Introduction

On couvre ici les principaux problèmes rencontrés ainsi que les solutions envisagées et explorées puis celles adoptées. Les problèmes ont été observés dans différentes parties du projet de recherche et ont affecté l'atteinte de différents objectifs.

4.2 Problèmes rencontrés

Outre les problèmes normaux de déverminage survenus lors de l'implantation et des simulations, des problèmes de nature plus théorique ont fait surface. Cinq d'entre eux sont traités en particulier.

- Le premier provient du fait que plusieurs règles de combinaison ne respectent pas la propriété mathématique d'associativité. L'ordre d'arrivée des informations a donc une importance. On remarque que naturellement les dernières informations reçues affectent davantage que les précédentes.
- Le second problème est que la règle de combinaison adaptative (ACR) qui devait s'étendre sous le cadre de raisonnement D^{Θ}, ne l'est pas en réalité.
- Le troisième problème concerne l'algorithme de quasi-associativité. Ce dernier utilise les règles conjonctive et disjonctive en séquence. Ce faisant, il accumule sans cesse les masses attribuées vers les ensembles d'intersection totale et d'union totale sans pour autant les redistribuer. Le résultat obtenu est alors toujours le même selon la méthode de distribution de la règle de combinaison qui se base seulement sur les ensembles d'intersection totale ou d'union totale. Sans quoi, il faut renoncer à l'associativité.
- Le quatrième problème provient de la nature sensible de la fonction de masse résultant de la combinaison. Cela rend les décisions difficiles à prendre. Distinguer entre une erreur temporaire sur l'information ou un changement d'allégeance devient d'autant plus complexe. Ce problème rend la prise de décision peu fiable.
- Le cinquième et dernier problème est issu de la transformation pignistique généralisée. Une certaine incohérence logique dans la méthodologie jette un doute sur la valeur des résultats obtenus.

4.3 Problème avec la propriété d'associativité

Le non-respect de la propriété d'associativité pour une règle de combinaison rend l'ordre de combinaison des informations important. Les informations passées sont moins influentes que les plus récentes. Ainsi, une règle de combinaison non associative devient très sensible à l'arrivée de contre-mesures ou d'informations erronées. Ainsi, un décideur basant sa décision sur la sortie d'une règle de combinaison non associative peut en arriver à un mauvais jugement à cause d'une mauvaise information survenue dans les dernières combinaisons.

La figure 4.1 montre un exemple[1] avec la règle de combinaison adaptative symétrique (SACR) utilisée sur des informations choisies parmi l'ensemble $\{\theta_1, \theta_2, \theta_3\}$, où θ_1 est sélectionné aléatoirement 8 fois sur 10 durant les 50 premières itérations, suivi de θ_3 8 fois sur 10 pour les 50 itérations suivantes. Les objets θ_2 et θ_3 se partagent également le reste de la probabilité d'occurrence dans les 50 premières itérations et de façon similaire avec les objets θ_1 et θ_2 dans les 50 dernières itérations. À chaque itération, on attribue une masse de 0.3 à l'ignorance totale, et de 0.7 sur le singleton sélectionné.

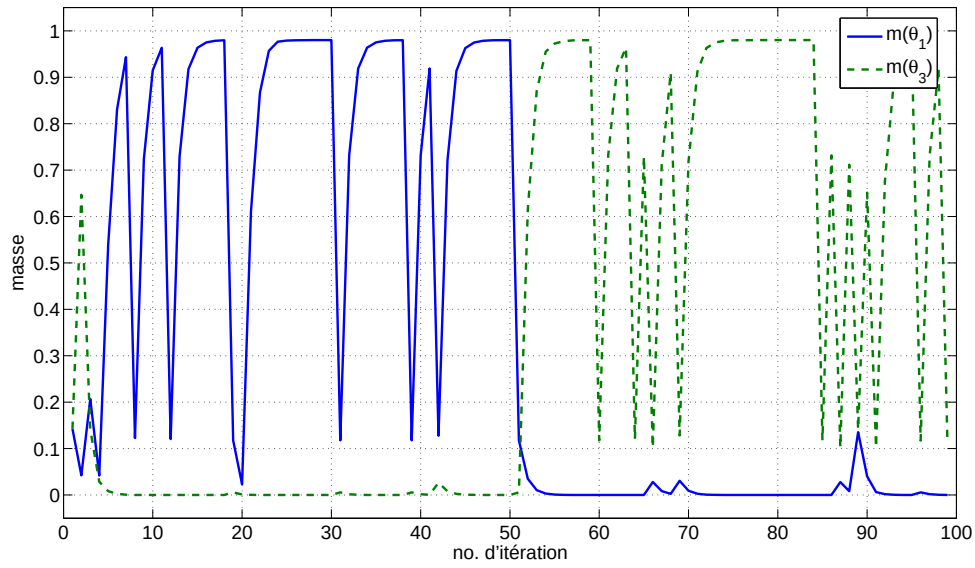


FIG. 4.1 – Fusion dans un contexte non associatif comportant plusieurs contre-mesures ou informations erronées

[1]Notez que la décision ne se prend pas en se basant sur la fonction de masse. Cet exemple démontre tout de même le fort niveau d'oscillation de cette fonction de masse qui pourra se répercuter, entre autres, sur la probabilité pignistique.

Cependant, cette non-associativité de plusieurs règles de combinaisons ne doit pas toujours être considérée comme étant désavantageuse. Dans certains contextes qui évoluent avec le temps, il est normal de considérer plus fortement les informations les plus récentes, ce que réalisent les règles non associatives.

4.3.1 Quasi-associativité

Une manière d'atteindre la quasi-associativité, selon [13] est présentée à la figure 3.5. L'auteur de [36] a également travaillé sur cette manière d'atteindre la quasi-associativité. Même si l'associativité est bien définie sous le modèle de Shafer, dans l'ensemble de puissance 2^Θ, des adaptations sont requises pour implantation sous le modèle hybride de DSmT dans l'ensemble d'hyperpuissance D^Θ. La section 4.4 du présent chapitre porte sur le problème que l'on rencontre avec la quasi-associativité.

4.3.2 Associativité locale

L'associativité locale est une autre manière de contourner la non-associativité. L'idée consiste à générer, à partir de la séquence originale des résultats de combinaisons, toutes les séquences possibles à partir de toutes les permutations des sorties de combinaison. Puis, on procède à l'aide d'une moyenne de tous les résultats des informations au temps τ.

Étant donnée l'explosion combinatoire et la volonté d'obtention d'un système qui fonctionne en temps réel, cette manière nécessite un fenêtrage. C'est-à-dire que l'on doit délimiter une période de temps sur laquelle on cumulera toutes les sorties pour évaluer la moyenne arithmétique de toutes les permutations possibles sur cette fenêtre de temps choisie. La propriété de commutativité que respectent les règles de combinaisons permet d'éviter l'évaluation de toutes les permutations donnant les mêmes résultats. On remarque toutefois qu'éviter toutes les possibilités identiques issues de la commutativité, risque de fausser le résultat moyen si on ne tient pas compte du nombre d'occurrence de possibilités dédoublées.

Dans le cadre de notre recherche, nous avons minimisé le fenêtrage à une dimension de trois itérations. Pour ce cas, il y a 6 combinaisons de séquences de sortie à considérer pour le calcul.

4.3.3 Problème de l'associativité locale

L'associativité locale, telle que définie, n'est associative que pour la fenêtre sur laquelle elle est construite. Ce n'est donc pas de la quasi-associativité sur l'ensemble des informations, mais bien seulement sur une période prédéfinie. Malheureusement, le système avec associativité locale ne peut être exploité en temps réel que sur de petites fenêtres étant donnée la complexité croissante en fonction de la taille de la fenêtre.

4.4 Problème de l'ACR sous la DSmT

La règle de combinaison créée par M. Florea [13], dont une version est acceptée pour publication dans *Information Fusion*, est définie pour un espace de raisonnement sous 2^Θ. L'auteur de cette règle mentionne, comme plusieurs le font pour leurs règles [26], qu'une extension est possible sous D^Θ, sans plus de détails, expérimentations ou exemples. Dans le cas de notre étude, une transposition sous D^Θ était nécessaire pour toutes les règles qui le permettaient. Hélas, on a ainsi démontré que l'ACR ne permet pas la transposition directe sous sa définition actuelle. En effet, sous D^Θ, la possibilité qu'un ensemble produit par une disjonction soit conflictuel existe. Ce conflit disjonctif n'est pas géré par l'ACR qui n'est alors pas en mesure de faire correctement une combinaison.

Améliorations possibles

À toutes règles de combinaison, divers raffinements sont possibles afin de les améliorer. On propose ici de répartir une part du conflit sur des ignorances partielles même si cela n'a pas été testé durant nos recherches. Ainsi, on tente logiquement de redistribuer une part des masses du conflit à l'ignorance partielle plutôt que de redistribuer ces masses directement à un niveau d'ignorance plus élevé.

4.4.1 Conflit disjonctif

L'établissement d'un nouveau type conflit est requis pour permettre une résolution de la non-existence d'une disjonction. Il a déjà été utilisé implicitement par [25] lorsqu'il a défini l'espace D^Θ. Une définition explicite est présentée ici. Cela permettra de définir un système de résolution mathématique d'une nouvelle combinaison dérivée de l'ACR.

Conflit disjonctif

On définit le conflit disjonctif $K_{\cup}$ comme étant le conflit sur un ensemble issu d'une disjonction, par opposition au conflit conjonctif qui, quant à lui, a lieu sur un ensemble issu d'une conjonction. Ce dernier a été vu à la section 2.3.3. Plus explicitement, le conflit $K_{\cup}$ survient lorsque des unions contraintes obtiennent une masse lors de l'opération de disjonction. Il est défini mathématiquement par :

$$K_{\cup} = \sum_{A \cup B = \emptyset} m_1(A)\, m_2(B) \qquad A, B \subseteq \Theta. \tag{4.1}$$

4.4.2 EACR, une ACR étendue sous DSmT

Deux faits marquent le passage de la DST vers la DSmT :
- il est maintenant possible d'avoir des masses sur les ensembles conjonctifs ;
- une démarcation entre la DSmT de la DST provient de la manière de répartir la masse conflictuelle.

Sous la DSmH, on répartit la masse conflictuelle aux ensembles qui causent le conflit, en opposition à la règle de Dempster qui répartit la masse du conflit sans cette considération. Dans la mesure où la règle de Dempster renvoie le conflit de manière globale à tous les éléments focaux, alors que la DSmH vise d'abords les ensembles ayant pris part au conflit. Cette différence dans la répartition de la masse du conflit est à la source de la non-associativité d'une règle comparativement à une autre.

L'ACR quant à elle, se démarque par un fait original par rapport à la combinaison de Dempster. La règle ACR calcule à la fois la conjonction et la disjonction lors de la combinaison, mais tient compte de l'une ou de l'autre par le biais d'un facteur de pondération fonction du niveau de conflit conjonctif. Autrement la masse du conflit conjonctif demeure répartie au reste des éléments focaux comme le fait la règle de Dempster.

Pour la création de la EACR, nous avons appliqué les mêmes raisonnements mais sous DSmT. La EACR a la particularité de tenir compte du conflit conjonctif ainsi que du conflit disjonctif. C'est ainsi que nous sommes partis de la règle ACR pour la remodeler et développer la EACR. Cette dernière pondère non seulement les conjonctions et disjonctions, mais permet le renvoi d'une masse résiduelle à l'ignorance totale en fonction du niveau de conflit disjonctif selon certaines conditions sur le conflit conjonctif.

Équations et conditions de fonctionnement de la EACR

La Règle de Combinaison Adaptative Étendue (EACR) s'adapte pour donner plus de poids aux résultats de combinaisons conjonctives, disjonctives, ou sur l'ignorance totale selon des règles de pondération basées sur le conflit conjonctif ainsi que le conflit disjonctif. La EACR ne permet pas de masse à l'ensemble vide et procède ainsi à la combinaison :

$$m_{EACR}(A) = \mu(K_\cap, K_\cup) m_\vee(A) + \eta(K_\cap) m_\wedge(A) + \rho(K_\cap, K_\cup) \tag{4.2}$$

où

$$\rho(K_\cap, K_\cup) \text{ est une fonction de } (\mu(\cdot), \eta(\cdot)) \tag{4.3}$$
$$\mu(K_\cap, K_\cup) \to 0 \text{ lorsque } K_\cap \to 1 \tag{4.4}$$
$$\mu(K_\cap, K_\cup) \to 0 \text{ lorsque } K_\cap \neq 0 \text{ et } K_\cup = 1 \tag{4.5}$$
$$\mu(K_\cap, K_\cup) \to 1 \text{ lorsque } K_\cap = 1 \text{ et } K_\cup = 0 \tag{4.6}$$
$$\mu(K_\cap, K_\cup) \text{ pour } K_\cup \text{ fixe, croît à mesure que } K_\cap \text{ croît.} \tag{4.7}$$
$$\mu(K_\cap, K_\cup) \text{ pour } K_\cup \text{ fixe, décroît à mesure que } K_\cap \text{ décroît.} \tag{4.8}$$
$$\mu(K_\cap, K_\cup) \text{ pour } K_\cap \text{ fixe, croît à mesure que } K_\cup \text{ décroît.} \tag{4.9}$$
$$\mu(K_\cap, K_\cup) \text{ pour } K_\cap \text{ fixe, décroît à mesure que } K_\cup \text{ croît.} \tag{4.10}$$
$$\eta(K_\cap) \text{ décroît à mesure que } K_\cap \text{ croît.} \tag{4.11}$$
$$\eta(K_\cap) \text{ croît à mesure que } K_\cap \text{ décroît.} \tag{4.12}$$
$$\eta(K_\cap) \to 0 \text{ lorsque } K_\cap = 1 \tag{4.13}$$
$$\eta(K_\cap) \to 1 \text{ lorsque } K_\cap = 0 \tag{4.14}$$

Toutes fonctions $\mu(\cdot)$ et $\eta(\cdot)$ qui respectent les propriétés présentées aux équations (4.3) à (4.14) sont éligibles. Cependant, il est encore plus intéressant de choisir des fonctions qui gèrent bien les dérivées secondes. Dans le sens où les fonctions choisis offre, entre autres, un contrôle sur la pente de la pente de la courbe. Nous avons alors sélectionné des fonctions de $\mu(\cdot)$ et de $\eta(\cdot)$ qui comportent des exponentielles.

$$\eta(K_\cap) = \left[2^{(-1)(2K_\cap)} \left(\frac{4}{3}\right) - \frac{1}{3}\right] \tag{4.15}$$

$$\mu(K_\cap, K_\cup) = 1 - \left[2^{(-1)(2K^*_\cup + K_\cap)} \left(\frac{4}{3}\right) - \frac{1}{3}\right] \tag{4.16}$$

$$\rho(K_\cap, K_\cup) = [1 - (1 - K^*_\cup)\mu(K_\cap, K_\cup) - (1 - K_\cap)\eta(K_\cap)] \tag{4.17}$$

où

$$K^*_\cup = K_\cup \quad \forall K_\cap \neq 0 \tag{4.18}$$

$$K^*_\cup = 0 \quad \forall K_\cap = 0 \tag{4.19}$$

Fonctionnement et expérimentations avec la règle EACR

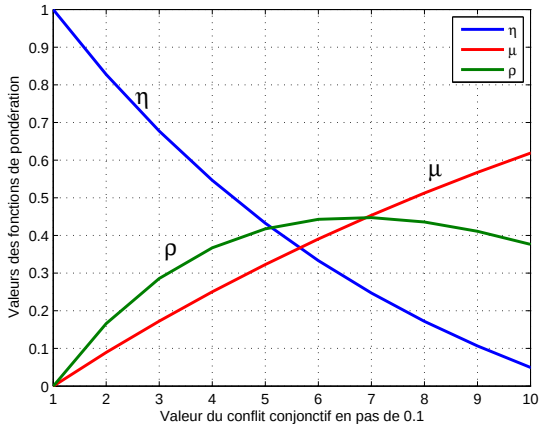
La présente section présente le fonctionnement pratique de la règle de combinaison EACR. On présente trois figures (voir fig. 4.2) montrant le comportement des fonctions de pondération μ, η et ρ en fonction du conflit conjonctif pour des cas ayant des conflits disjonctifs à 0, 0.5, puis de 0.8. On présente également pour chacun des cas développés, les réponses que donnent d'autres règles de combinaison pour de mêmes entrées.

Exemple 4.1 Cas simple de combinaison par la règle EACR

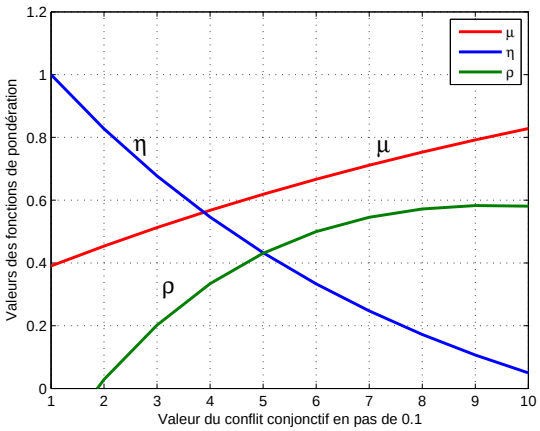
Ce premier exemple d'un cas simple de combinaison par la règle EACR comporte deux sources d'informations, raisonnant sous D^Θ *avec un cardinal de* 2*. C'est-à-dire qu'il n'y a que deux singletons possibles. Les deux informations produisent un fort niveau de conflit conjonctif s'élevant à* 0.45*.*

- *Le capteur* 1 *donne l'information suivante :*
  $\{m(A) = 0.70; m(B) = 0.10; m(A \cup B) = 0.20\}$.
- *Le capteur* 2 *donne l'information suivante :*
  $\{m(A) = 0.30; m(B) = 0.60; m(A \cup B) = 0.10\}$.

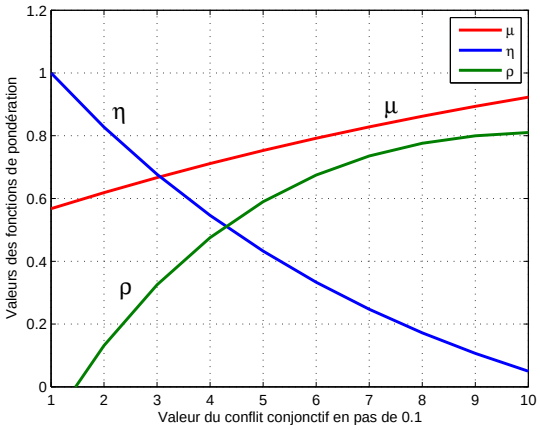
La table 4.3 montre que toutes les règles de combinaison misent 2 fois plus sur l'objet A que B. Par contre, les règles SACR, DSmH, EACR accordent plus d'importance



4.2.1: Cas où le conflit disjonctif est de 0.0



4.2.2: Cas où le conflit disjonctif est de 0.5



4.2.3: Cas où le conflit disjonctif est de 0.8

FIG. 4.2 – Comportement des fonctions de pondérations μ, η et ρ en fonction de $K_{\cap}$

TAB. 4.1 – Produits des intersections et unions

	$m_1(A) = 0.7$	$m_1(B) = 0.1$	$m_1(A \cup B) = 0.2$
$m_2(A) = 0.3$	$m_\wedge : A$ 0.21 $m_\vee : A$	$m_\wedge : A \cap B$ 0.03 $m_\vee : A \cup B$	$m_\wedge : A$ 0.06 $m_\vee : A \cup B$
$m_2(B) = 0.6$	$m_\wedge : A \cap B$ 0.42 $m_\vee : A \cup B$	$m_\wedge : B$ 0.06 $m_\vee : B$	$m_\wedge : B$ 0.12 $m_\vee : A \cup B$
$m_2(A \cup B) = 0.1$	$m_\wedge : A$ 0.07 $m_\vee : A \cup B$	$m_\wedge : B$ 0.01 $m_\vee : A \cup B$	$m_\wedge : A \cup B$ 0.02 $m_\vee : A \cup B$

TAB. 4.2 – Détails des calculs de masses d'intersections, d'unions et des conflits

A	$0.21 + 0.06 + 0.07$	$m_\wedge(A) = 0.34$
B	$0.12 + 0.06 + 0.01$	$m_\wedge(B) = 0.19$
$A \cup B$	0.02	$m_\wedge(A \cup B) = 0.02$
$K_\cap$	$0.03 + 0.42$	$K_\cap = 0.45$
A	0.21	$m_\vee(A) = 0.21$
B	0.06	$m_\vee(B) = 0.06$
$A \cup B$	$0.06 + 0.12 + 0.07 + 0.01 + 0.02$	$m_\vee(A \cup B) = 0.28$
$K_\cup$	0.00	$K_\cup = 0.00$

absolue à I_t. Seules les règles PCR5 et DS risquent un choix[2] *vers l'objet A.*

Exemple 4.2 Second cas de combinaison par la règle EACR

Cet exemple de combinaison par la règle EACR comporte également deux sources d'informations raisonnant sous D^Θ avec un cardinal de 2. Toutefois, la différence se situe au niveau du conflit, plus faible qu'à l'exemple 4.1, se situant cette fois-ci à un niveau de 0.30.

- *Le capteur 1 donne l'information suivante :*
  $\{m(A) = 0.80; m(B) = 0.10; m(A \cup B) = 0.10\}$.
- *Le capteur 2 donne l'information suivante :*
  $\{m(A) = 0.60; m(B) = 0.30; m(A \cup B) = 0.10\}$.

La table 4.4 montre que toutes les règles de combinaison mettent entre 6 et 10 fois

[2]Notez qu'une décision dans un cas réel ne se prendra pas à partir de la masse.

TAB. 4.3 – Résultat de combinaisons sous diverses règles de combinaison pour l'exemple 4.1

	Dempster	SACR	EACR	DSmH	PCR5
$m(A)$	0.6182	0.3741	0.2046	0.3400	0.5887
$m(B)$	0.3455	0.1748	0.0939	0.1900	0.3913
$m(A \cup B)$	0.0364	0.4512	0.7015	0.4700	0.0200

TAB. 4.4 – Résultat de combinaisons sous diverses règles de combinaison pour l'exemple 4.2

	Dempster	SACR	EACR	DSmH	PCR5
$m(A)$	0.8857	0.7316	0.4589	0.6200	0.846
$m(B)$	0.1000	0.0734	0.0458	0.0700	0.1440
$m(A \cup B)$	0.0143	0.1949	0.4954	0.3100	0.0100

plus de poids sur l'objet A que sur l'objet B, par contre, la règle EACR accorde plus d'importance absolue à l'ignorance totale. Toutes les autres règles permettent le choix de l'objet A.

Exemple 4.3 Comportement de la EACR avec le problème de Zadeh modifié

Cet exemple de combinaison par la règle EACR comporte également deux sources d'informations raisonnant sous D^Θavec un cardinal de 3. Toutefois, on reproduit partiellement[3] *ici un cas ayant une opposition élevée entre les deux sources pour s'approcher du problème de Zadeh. On connaît déjà le comportement que devrait avoir la règle de Dempster : la masse sera mise sur le singleton qu'aucune des deux sources ne préconise. Toutefois, nous n'avons pas exploré ce cas avec les autres règles jusqu'à présent.*

- *Le capteur* 1 *donne l'information suivante :*
  $\{m(A) = 0.50; m(C) = 0.20; m(A \cup C) = 0.10; m(A \cup B \cup C) = 0.20\}$.
- *Le capteur* 2 *donne l'information suivante :*
  $\{m(B) = 0.50; m(C) = 0.20; m(B \cup C) = 0.10; m(A \cup B \cup C) = 0.20\}$.

La table 4.5 renseigne sur plusieurs comportements intéressants qu'ont les règles de

[3] Ce n'est pas exactement le cas avec les mêmes données que le problème de Zadeh, mais un cas équivalent où une source mise la grande majorité de la masse sur un objet, et où la seconde source mise la grande majorité de la masse sur un autre objet.

TAB. 4.5 – Résultat de combinaisons pour un cas hautement conflictuel

	DST	SACR	EACR	DSmH	PCR5
$m(A)$	0.2272	0.0584	0.0280	0.1000	0.3381
$m(B)$	0.2272	0.0584	0.0280	0.1000	0.3381
$m(C)$	0.36	0.1231	0.062	0.1600	0.2171
$m(A \cup B)$	-	0.1858	0.1072	0.2500	-
$m(A \cup C)$	0.045	0.1008	0.0571	0.1200	0.0200
$m(B \cup C)$	0.045	0.1008	0.0571	0.1200	0.0200
$m(A \cup B \cup C)$	0.09	0.3726	0.6606	0.1500	0.0400

combinaison explorées. D'abord, dans tous les cas, la masse attribuée au singleton A est identique à celle attribuée au singleton B dû à la symmétrie de l'exemple et de la commutativité. La règle de Dempster se comporte tel que prévu attribuant plus de masse au singleton C.

Les règles PCR5 et DSmH ont un comportement similaire mais une approche différente. Dans le cas de la DSmH, on dit que c'est $A \cup B$ qui est l'objet de plus grande importance. Par ce fait, on dit que c'est soit A, soit B, ayant ainsi une ignorance partielle. Dans le cas de la PCR5, on n'est pas en mesure d'en arriver à l'union : la règle reste bloquée, avec la majorité de la masse partagée également entre A et B ne permettant pas de prendre de décision[4]*. Il se trouve également une masse importante sur l'objet C.*

Les règles SACR et EACR s'apparentent entre elles. Elles misent toutes deux principalement sur l'ignorance totale, puis l'objet de seconde importance sur lequel est envoyé une masse importante est l'ignorance partielle qui partage le choix entre A et B.

Les règles SACR, EACR, et DSmH semblent donc avoir tendance à aller vers les ignorances partielles et totales permettant de prendre une juste décision plus tard tout en délimitant, par les ignorances partielles, le champ des possibilités subséquentes.

On peut déjà tirer quelques conclusions en ce qui a trait au comportement de la règle de combinaison EACR. Elle semble avoir tendance à privilégier[5] l'ignorance à la prise de position d'un singleton spécifique dans cet exemple. Ce n'est pas nécessairement un point négatif ou nuisible puisque comme il sera vu au chapitre suivant, la règle demeure

[4] On rappelle qu'en réalité la décision ne se prend pas sur la masse.

[5] Ce comportement est semblable à celui que l'on aurait sous la règle de Yager.

apte à prendre de bonnes décisions. Aussi, le fait qu'elle ait une forte tendance à aller vers l'ignorance en cas de conflit, lui permet de récupérer plus rapidement en cas de changement d'allégeance. Aussi, lors de doute, elle signale ce doute clairement dans sa réponse par le choix d'ignorance partielle ou totale, par opposition aux règles qui tentent tout de même un choix malgré la présence de doute.

4.5 Problème de la quasi-associativité

Lors d'expérimentations avec la quasi-associativité telle que décrite dans la littérature [13, 26], nous avons découvert que la règle de combinaison conjonctive accumule éventuellement toute la masse dans l'intersection totale. La combinaison disjonctive agit de même en accumulant la masse dans l'union totale à mesure que les conflits ont lieu. Cela était prévisible. Ceci empêche évidemment les règles de combinaison à être quasi-associatives. On note également que plus les masses aux objets en conflits sont élevées, plus le transfert de masses vers l'intersection totale, ou l'union totale, est rapide.

4.5.1 Accumulation vers l'intersection totale

L'exemple 4.4 montre, pour un simple cas de règle conjonctive, comment la masse se trouve dirigée vers l'intersection totale $\cap_T$.

Exemple 4.4 Accumulation vers l'intersection totale

Pour un cas sous D^Θ avec un cardinal du cadre de discernement de 3*, on applique une première fois la règle conjonctive*[6] *aux deux premières informations. Le résultat sera ensuite combiné à la troisième information.*

- *Le capteur* 1 *donne l'information suivante :*
  $\{m_1(\theta_1) = 0.2, m_1(\theta_1 \cap \theta_2) = 0.2, m_1(\theta_1 \cap \theta_2 \cap \theta_3) = 0.6\}$.
- *Le capteur* 2 *donne l'information suivante :*
  $\{m_2(\theta_3) = 0.7, m_2(\theta_1 \cup \theta_2 \cup \theta_3) = 0.3\}$.
- *Le capteur* 3 *donne l'information suivante :*
  $\{m_3(\theta_2) = 0.7, m_3(\theta_1 \cup \theta_2 \cup \theta_3) = 0.3\}$.

[6] Il peut également s'agir de toute autre règle à base conjonctive sans application de contraintes.

La première combinaison donne le résultat suivant :

$$\begin{aligned} m_{12}(\theta_1) &= 0.2 \times 0.3 = 0.06 \\ m_{12}(\theta_1 \cap \theta_2) &= 0.06 \\ m_{12}(\theta_1 \cap \theta_2 \cap \theta_3) &= 0.74 \\ m_{12}(\theta_1 \cap \theta_3) &= 0.14 \end{aligned}$$

La seconde combinaison donne le résultat suivant :

$$\begin{aligned} m_{123}(\theta_1) &= 0.3 \times 0.06 = 0.018 \\ m_{123}(\theta_1 \cap \theta_2) &= 0.7 \times 0.06 + 0.7 \times 0.06 + 0.06 \times 0.3 = 0.102 \\ m_{123}(\theta_1 \cap \theta_2 \cap \theta_3) &= 0.7 \times 0.14 + 0.7 \times 0.74 + 0.74 \times 0.3 = 0.838 \\ m_{123}(\theta_1 \cap \theta_3) &= 0.3 \times 0.14 = 0.042 \end{aligned}$$

On voit donc que s'il y a des contre-mesures, de sorte que le singleton identifié varie faiblement, et $\cap_T$ prend rapidement de l'ampleur sans laisser de chances de récupération. On remarque même qu'après la première combinaison, le singleton identifié par le second corps d'évidence soit θ_3, n'est pas présent parmi les résultats. Il en est de même pour θ_2 après la seconde combinaison, malgré son identification par la troisième information ; θ_2 est simplement absent du résultat de combinaison.

4.5.2 Solution par seuillage sur l'ignorance totale

Une solution simple au problème d'accumulation de masse vers $\cap_T$ est d'appliquer un seuil sur le niveau minimal de la masse à assigner à l'ignorance totale à chaque combinaison d'informations. Un certain succès est ainsi obtenu, car on donne la possibilité de générer de l'information autre que l'intersection totale.

Exemple 4.5 Accumulation vers $\cap_T$ en considérant un seuil sur I_t

On reprend ici les données de l'exemple 4.4, où un seuil de 0.05 *est appliqué à I_t à chaque étape de combinaison. Après la première combinaison, on a donc :*

$$\begin{aligned} m_{12}(\theta_1) &= (1 - 0.05)(0.06) = 0.057 \\ m_{12}(\theta_1 \cap \theta_2) &= (0.95)(0.06) = 0.057 \\ m_{12}(\theta_1 \cap \theta_2 \cap \theta_3) &= (0.95)(0.74) = 0.703 \\ m_{12}(\theta_1 \cap \theta_3) &= (0.95)(0.14) = 0.133 \\ m_{12}(\theta_1 \cup \theta_2 \cup \theta_3) &= 0.05 \end{aligned}$$

Après la seconde combinaison, on obtient :

$$\begin{aligned} m_{123}(\theta_1) &= 0.0171 \\ m_{123}(\theta_1 \cap \theta_2) &= 0.0969 \\ m_{123}(\theta_1 \cap \theta_2 \cap \theta_3) &= 0.7961 \\ m_{123}(\theta_1 \cap \theta_3) &= 0.0399 \\ m_{123}(\theta_1 \cup \theta_2 \cup \theta_3) &= 0.0150 \\ m_{123}(\theta_2) &= 0.0350 \end{aligned}$$

Résultat sur lequel on doit également appliquer le filtre permettant d'assurer un seuil minimal à I_t, on obtient donc :

$$\begin{aligned} m_{123}(\theta_1) &= (1-(0.05-0.0150))(0.0171) = 0.0165 \\ m_{123}(\theta_1 \cap \theta_2) &= (0.965)(0.0969) = 0.0935 \\ m_{123}(\theta_1 \cap \theta_2 \cap \theta_3) &= (0.965)(0.7961) = 0.7678 \\ m_{123}(\theta_1 \cap \theta_3) &= (0.965)(0.0399) = 0.0385 \\ m_{123}(\theta_1 \cup \theta_2 \cup \theta_3) &= 0.0500 \\ m_{123}(\theta_2) &= (0.965)(0.0350) = 0.0338 \end{aligned}$$

Par l'exemple 4.5, on voit que la présence d'un seuil minimal sur I_t permet l'obtention d'ensembles focaux nouveaux. Ainsi, cette fois, on peut obtenir une certaine masse au singleton θ_2, par exemple. Noter également que l'impact sur les masses des différents ensembles est différent. Le filtre assure un minimum à I_t, mais ce dernier trouve cette masse en la prenant, parmi tous les ensembles focaux, proportionnellement à la masse qui leur est attribuée.

4.5.3 Redistribution du conflit considérant la fiabilité

Une autre des solutions envisagées est de redistribuer partiellement ou en totalité une partie des masses du conflit selon la fiabilité des sources impliquées vers la masse attribuée à I_t.

Il serait également possible de considérer l'apport de l'information d'une source en fonction de sa fiabilité. Une telle approche se présenterait comme suit. Soit γ_A, le niveau de fiabilité d'un capteur A et γ_B celle d'un capteur B. Une normalisation sera sans doute nécessaire, ce que le facteur de normalisation Ψ permettra de réaliser.

Lors d'une combinaison d'informations provenant de ces deux capteurs, on effectue l'opération décrite par l'équation[7] suivante :

$$(m_A \odot m_B)(C) = \Psi \sum_{X \cap Y = C} (\gamma_A(m_A)\, m_A(X))(\gamma_B(m_B)\, m_B(Y)) \qquad \forall X, Y \in D^\Theta \tag{4.20}$$

Les fonctions de pondération γ_A et γ_B de quantification de la fiabilité des sources A et B pourraient être vues comme des fonctions avec les critères de définition de fonction suivantes :

- doivent tendre asymptotiquement vers un seuil de fiabilité maximal,
- doivent devenir nul et donc rejeter l'information d'une source lors de l'atteinte d'un seuil inférieur,
- divers tests et simulations sous divers contextes pourraient aider à optimiser ces seuils suivant la volonté du commandement quant aux qualités du système,
- doivent être fonction du taux de bonnes décisions prises récemment[8],
- doivent posséder un comportement logique graduel ; mathématiquement, la fonction doit être dérivable, continue et monotone,
- doivent également répondre aux principes de la théorie des probabilités puisque la fiabilité consiste de fait en une application de la théorie des probabilités aux problèmes de durée de fonctionnement sans incident[9].

On retrouvre un développement similaire dans [4].

4.5.4 Quasi-associativité avec DS

La solution choisie pour atténuer le problème d'accumulations vers $\cap_T$ et $\cup_T$ des règles de base pour la quasi-associativité, a été :

- de remplacer la conjonction par la règle de Dempster (DS),
- de remplacer la disjonction par une version disjonctive (DSTd) de la règle de Dempster. La DSTd est présenté à la prochaine section.

Le schéma de l'algorithme apparaissant à la figure 3.5 a été refait pour tenir compte des conflits conjonctif et disjonctif en utilisant les règles de Dempster et la nouvelle DSTd. Le tout apparaît sur la figure 4.3.

[7] L'équation devrait également inclure un facteur de normalisation pour respecter la définition de fonction de masse.

[8] un évènement imprévu ayant pu avoir lieu perturbant la fiabilité d'une source

[9] Dans notre cas, cela correspondrait plutôt à la durée de fonctionnement sans erreur de décision.

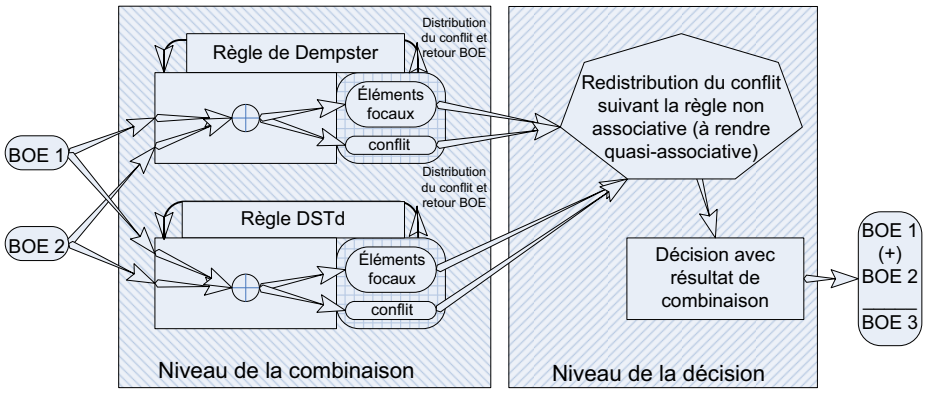


FIG. 4.3 – Algorithme de quasi-associativité avec DST et DSTd

4.5.5 DST sous forme disjonctive (DSTd)

La DST sous forme disjonctive (DSTd) contient l'adaptation que nous avons apportée à la DST pour lui permettre de travailler à base de disjonctions ainsi qu'avec le conflit disjonctif tel que défini à la section 4.4.1. Il ne sert à rien de tenter d'y trouver une correspondance philosophique, car cette règle n'est qu'un outil mathématique théorique permettant d'établir une correspondance à l'ACR sous l'ensemble d'hyperpuissance. La règle de combinaison DSTd est une règle disjonctive normalisée fonctionnant sur l'ensemble de puissance $2^{\Theta}_{\cup}$. Où $2^{\Theta}_{\cup}$ est l'ensemble de puissance tel que l'élément $\{\theta_i, \theta_j\}$ correspond à $\{\theta_i \cap \theta_j\}$ et non $\{\theta_i \cup \theta_j\}$ dans 2^{Θ}. La DSTd combine les informations avec des unions uniquement. L'équation (4.21) décrit la règle de combinaison DSTd où $K_{\cup}$ représente le conflit disjonctif.

$$(m_1 \bar{\oplus} m_2)(C) = \frac{1}{1 - K_{\cup}} \sum_{A \cup B = C} m_1(A)\, m_2(B) \qquad \forall C \subseteq \Theta \tag{4.21}$$

4.6 Problème d'oscillation de la fonction de masse

Les combinaisons d'informations ont tendance à fournir des fonctions de masse sensibles qui conduisent à des masses oscillant entre deux ou plusieurs valeurs.

En effet, l'apparition de seulement deux ou trois entrées consécutives erronées suffit pour que certaines règles de combinaison modifient les fonctions de masse significativement. Les décisions prises sur masse maximale alternent entre objets. Une façon d'éviter

un changement ou une alternance de la décision est d'appliquer un filtrage.

4.6.1 Création et utilisation d'un filtre sur les sorties

Les propriétés désirées d'un filtre sur les sorties sont plutôt simples. On souhaite un filtre léger au niveau arithmétique, qui atténue les oscillations de la courbe de la masse d'un objet en fonction du temps tout en maintenant un résultat cohérent. Un filtre élémentaire, que nous avons exploré, consiste à prendre la moyenne arithmétique de la sortie courante avec la moyenne des k combinaisons précédentes. Cela correspond donc à l'équation suivante :

$$m_{\tau_f} = \left(\alpha m_\tau + \beta \left(\frac{m_{\tau-3} + m_{\tau-2} + m_{\tau-1}}{3}\right)\right)/2 \tag{4.22}$$

où α et β valent 1 ; toutefois nous avons aussi appliqué une pondération variable qui donne plus de pondération à la valeur courante si la différence entre cette dernière et la moyenne des trois valeurs précédentes est petite. Logiquement, plus de pondération est accordée au passé si la différence est plus grande.

Les filtres sur les sorties risquent cependant de donner un caractère temporel aux décisions d'où un comportement non associatif. Ce type de filtre n'est donc pas recommandé.

4.6.2 Création et utilisation d'un filtre sur les entrées

Un raisonnement plus poussé sur les objectifs désirés d'un filtre, ainsi que les effets indésirables produits par les filtres sur les sorties, nous ont amené à revoir les grands principes et trouver les causes potentielles de la sensibilité[10] des règles de combinaison. Les réflexions ont permis de dégager les qualités recherchées par une règle de combinaison :

- un résultat représentatif de la réalité selon les informations obtenues,
- un résultat en mesure de changer relativement rapidement lorsque l'information se précise pour conduire à un véritable changement de décision.

[10]Une forte sensibilité est parfois un préalable pour une règle de combinaison dans certains contextes où une réaction rapide est avantageuse.

De là émerge l'idée de constituer un filtre agissant plutôt sur les informations à combiner, non pas sur les sorties.

Avec ce filtre, un traitement de l'information a lieu avant la combinaison. Pour un relevé ESM au temps τ_T, on analyse les relevés ESM passés sur une fenêtre de temps, trois derniers relevés ESM par exemple. On a donc les relevés τ'_{T-3}, τ'_{T-2}. τ'_{T-1} et τ_T, où le symbole $'$ dénote un temps où le relevé ESM a été filtré.

- On vérifie en premier lieu l'identité (singleton) de la cible selon le capteur ESM au temps τ_T.
- Ensuite, on comptabilise le nombre d'occurrences de ce singleton parmi les relevés ESM de la fenêtre en cours d'analyse.
- Plus le ratio d'occurrence du singleton du relevé ESM au temps τ_T par rapport à la dimension de la fenêtre diminue, plus il faudra diminuer le niveau de certitude (masse) du singleton pour le relevé ESM[11] du temps τ'_T.

Pour notre étude nous avons affecté de la façon suivante le niveau de certitude du relevé ESM du temps τ_T, en fonction du nombre d'occurrences du singleton détecté au temps τ_T parmi les relevés ESM de la fenêtre surveillée :

- 4 occurrences sur une fenêtre de 4 relevés ESM : 70% au singleton détecté à τ'_T.
- 3 occurrences sur une fenêtre de 4 relevés ESM : 60% au singleton détecté à τ'_T.
- 2 occurrences sur une fenêtre de 4 relevés ESM : 50% au singleton détecté à τ'_T.
- 1 occurrences sur une fenêtre de 4 relevés ESM : 40% au singleton détecté à τ'_T.

On rappelle que par défaut, sans le filtre, on attribue une certitude de 70% au singleton détecté par un capteur ESM, et le 30% qui reste est attribué à l'objet I_t. Avec le filtre, on ne change pas le singleton rapporté, mais le niveau de certitude associé à sa détection. Ainsi, lorsqu'il y a un rapport singulier de θ_2, par exemple, parmi une séquence continue de θ_1, et donc probablement attribuable à une erreur, on diminue le niveau de certitude du relevé associé à θ_2 jusqu'à 40% de la masse totale ; ainsi I_t se voit attribuer la masse de 60%. Bien entendu, un effet secondaire de ce filtrage est qu'un vrai changement d'allégeance prendra plus de temps à se faire sentir. Il prendra jusqu'à 3 itérations supplémentaires dans le cas de notre étude pour les cas utilisant ce filtre.

[11] On rappelle que τ'_T correspond à la version filtrée du temps τ_T.

4.7 Problème avec la TPG

Une incohérence à été relevée dans les résultats de la TPG implanté tel que présenté dans [25]. La première cause possible de ce problème serait le fait que la masse n'est pas répartie selon la hiérarchie[12]. Selon cette manière de faire, on comptabilise les masses des objets reliés par hiérarchie[13] à l'objet dont on souhaite mesurer la probabilité pignistique au lieu de celles de tous les objets. L'exploration d'une TPG hiérarchique n'a pas été concluante. En effet, les versions potentielles de TPG hiérarchiques pensées ne permettaient pas le respect de tous les axiomes de la théorie des probabilités de Bayes. La seconde cause possible du problème est décrite au paragraphe subséquent.

4.7.1 Incohérence de la répartition des poids dans la TPG

L'incohérence de la répartition des poids dans la TPG[14] que l'on a relevée pour la TPG est en fait directement liée et causée par le cardinal DSm qu'il utilise. Ce n'est donc pas réellement un problème de la TPG mais du cardinal DSm. Ce problème est décrit à la section 4.7.2.

4.7.2 Cardinal DSm Généralisé

Le cardinal DSm, défini dans [25] et apparaissant dans les sections 2.5.2 et 2.5.4, attribue une valeur identique unitaire à chacun des ensembles possibles, qu'il soit inclus dans un autre, en totalité ou en partie, qu'il soit isolé, en totalité ou en partie, qu'il s'y trouve une petite ou une grande partie de la masse. Tous se retrouvent égaux dans le calcul du cardinal DSm. Cette valeur identique unitaire attribuée à chaque ensemble repose sur le cardinal d'un ensemble où chaque élément compte pour une valeur identique unitaire. Une valeur qui considère plusieurs facteurs tels ceux susmentionnées permettrait une plus grande flexibilité, d'où l'idée que l'on a eu d'un cardinal DSm

[12]Par hiérarchie, on veut dire de répartir la probabilité parmi les ensembles, sous-ensemble, et/ou ensemble-parent, etc.

[13]Un objet est hiérarchiquement relié à un autre lorsqu'ils sont membres d'ensembles parents ou enfants l'un par rapport à l'autre, ou plus loin dans la lignée familiale (ensemble-grand-parent, ensemble-parent, etc.) selon l'impact et la précision souhaitée.

[14]Du fait que la masse d'un est simplement répartie également entre ses parties lors de la conversion en probabilité pignistique sans considérer qu'il puisse y avoir plus de pondération à un sous ensemble plus qu'à un autre sous ensemble de l'ensemble dont on veut répartir la masse. (c.-à-d. $m(A)$ donne à part également sa masse à $P(A)$ et $P(A \cap B)$)

Généralisé.

En désirant respecter les axiomes et propriétés de la théorie de probabilité de Bayes une fois la transformation pignistique généralisée effectuée, les solutions possibles sont grandement restreintes. La seule solution explorée qui respecterait ces restrictions est cette généralisation du cardinal DSm qui attribue une valeur variable laquelle peut être calibrée par l'utilisateur du système au cas par cas selon l'usage.

Cette version du cardinal DSm est alors en mesure de maintenir la forme de la distribution de la certitude en fonction de la hiérarchie des objets à moins de calibration par l'usager l'en empêchant.

Exemple 4.6 Cardinal DSm vs Cardinal DSm Généralisé

Pour un cas où nous avons un cadre de discernement avec seulement deux singletons, A et B.

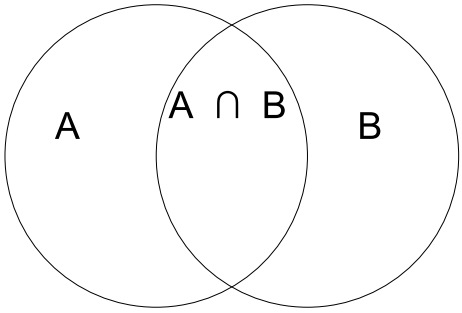


FIG. 4.4 – Représentation graphique d'un cas avec deux singletons

Avec le cardinal DSm on obtient :

$$\begin{aligned} C_{\mathcal{M}}(A) &= 2 \\ C_{\mathcal{M}}(B) &= 2 \\ C_{\mathcal{M}}(A \cap B) &= 1 \end{aligned}$$

où l'on a que les occurrences d'objet $\in A$ sont réparti à 50% dans $A \cap B$ et à 50% dans A. Similairement, pour les occurrences d'objet $\in B$, ils sont répartis à 50% dans $A \cap B$ et à 50% dans B. Mais la situation pourrait être telle que la répartition devrait être autrement.

Avec le cardinal DSm généralisé, on pourrait calibrer le niveau d'appartenance des

objets à un ensemble, tel que :

$$\begin{aligned} C_{\mathcal{M}}(A) &= 2.5 \\ C_{\mathcal{M}}(B) &= 1.5 \\ C_{\mathcal{M}}(A \cap B) &= 1 \end{aligned}$$

Ainsi, les objets $\in A$ *seraient répartis à* 40% *dans* $A \cap B$*. Pour un objet* $\in B$*, ils seraient répartis à* 66% *dans* $A \cap B$*.*

Comme le montre l'exemple 4.6, le cardinal DSm généralisé ouvre la porte à un cardinal DSm flou où l'appartenance d'objet θ_a à A ou aux ensembles qui l'intersecte serait gérée par la logique floue.

Cardinal DSm Généralisé Hiérarchique

En temps normal, c'est à dire pour le cas du cardinal DSm, toutes les possibilités d'ensembles sont mises au même pied d'égalité valant chacun une unité lors du calcul du cardinal et puis de la TPG. Le cardinal DSm hiérarchique généralisé est un cas de cardinal DSm généralisé où la distribution des proportions de masse se fait de façon hiérarchique, c'est-à-dire qu'un ensemble parent recevrait, une partie de la masse que l'ensemble enfant. La table 4.6 présente deux cas possibles de cardinal DSm généralisé hiérarchique. Notez que la notation des ensembles suit celle présentée à la figure 2.3. Les données de la table 4.6 ont été arbitrairement choisies pour montrer deux cas possibles où la répartition des parts s'est fait hiérarchiquement avec plus d'emphase[15] sur les singletons.

Tab. 4.6 – Deux cas possibles de cardinal DSm Généralisé Hiérarchique (CDGH)

Ensembles	1	2	3	12	13	23	123
Cardinal DSm	1	1	1	1	1	1	1
CDGH no.1	3	3	3	2	2	2	1
CDGH no.2	4	4	4	2	2	2	1

[15]En nombre de parts.

4.8 Conclusions

Le problème du non-respect de l'associativité a requis un certain travail de notre part pour amener au développement du concept d'associativité locale. Malgré tout, les résultats obtenus et montrés dans le prochain chapitre montrent que le problème reste entier.

Le problème de la quasi-associativité a conduit à la version disjonctive de la règle de Dempster afin de construire un algorithme quasi associatif utilisant la DST et la DSTd au lieu des règles conjonctive et disjonctive.

Le problème avec l'ACR a amené au développement d'une nouvelle règle, la EACR, qui fonctionne maintenant sous DSmT.

Le problème de l'oscillation entre plusieurs états de la décision après combinaisons d'informations erronées a entraîné l'emploi d'un filtre dont l'originalité consiste à l'appliquer au niveau des informations en affectant la masse attribuée à l'*allégeance esm* déclarée.

Enfin, le problème avec la TPG, et en fait, le cardinal DSm, a amené la suggestion d'une version hiérarchique qui n'a pas encore été testée.

Bien entendu, ces solutions, et diverses autres émises, mais non explorées en profondeur, sont à affiner et à explorer plus en détail comme le montreront les résultats du prochain chapitre. Elles constituent néanmoins une piste potentielle pour résoudre nos problèmes.

Chapitre 5

Résultats et analyse

Don't try to be a great man. Just be a man, and let history pass judgment. – Zefram Cochrane

5.1 Introduction

La présente recherche ne constitue pas une recherche exhaustive où sont comparées toutes les méthodes existantes de fusion d'informations. Comme mentionné précédemment, seules quelques règles choisies ainsi que quelques variantes de celles-ci ont été utilisées pour valider les performances et la qualité de la DSmH dans le contexte du STANAG 1241. On le rappelle, le STANAG 1241 définit la description de la structure de l'identification pour usage tactique. On y définit donc entre autres, la structure ou répartition de l'allégeance possible des cibles.

Ce que l'on désignait d'objet jusqu'à présent, représente indirectement l'*allégeance stanag* possible d'une cible qui peut être faite parmi les cinq choix suivants : ami ($\theta_1 - \theta_1 \cap \theta_2$), présumé ami ($\theta_1 \cap \theta_2$), neutre ($\theta_2 - \theta_1 \cap \theta_2 - \theta_2 \cap \theta_3$), suspect ($\theta_2 \cap \theta_3$)[1], et puis ennemi ($\theta_3 - \theta_2 \cap \theta_3$), ou, dans le cas des règles SACR et DS, l'*allégeance esm*. Aussi, tel que spécifié à la section 2.4.2, les capteurs ESM fournissent des informations quant à l'allégeance de la cible sous forme d'*allégeance esm*.

Le présent chapitre présente nos résultats[2] de simulations ainsi que notre analyse.

- La première partie présente les simulations ponctuelles étendues sur des périodes de cent unités temporelles pour les diverses règles de combinaisons présentées précédemment. Elles seront vues sous leur version originale, ainsi qu'avec des données filtrées, et puis dans quelques cas, utilisant la quasi-associativité.
- La seconde partie traite des simulations Monte-Carlo. Ces dernières procèdent à une centaine de simulations ponctuelles également de durée de cent unités de temps. Ces simulations se font pour diverses règles, avec une variété de calibration de paramètres agissant sur le filtre du seuil minimum à l'ignorance totale, le filtre sur les masses des entrées ainsi que sur la valeur du taux de détection de l'*allégeance esm*. On verra également le niveau de confiance que l'on peut apporter à nos valeurs par le biais de l'intervalle de confiance produit par nos simulations Monte-Carlo.

[1] Dans notre cas on considère la situation d'un pays en état de paix tel que montré à la figure 3.1.2

[2] Notez que pour des fins de simplification et pour une meilleure visualisation des résultats, seule les courbes ayant un niveau significatif sont présentées.

5.2 Résultats et analyse de simulations ponctuelles

Toutes les simulations de la présente section traitent l'ensemble des entrées ESM qui apparaissent à la figure 5.1. À chaque itération, le corps d'évidence consiste en un singleton θ_x correspondant à une *allégeance esm* d'une masse de 0.70, ainsi que de l'ignorance totale (I_t) qui obtient une masse de 0.30. On a également mis les ensembles $\theta_1 \cap \theta_3$ et $\theta_1 \cap \theta_2 \cap \theta_3$ sous contrainte.

Lors de la génération aléatoire des entrées, nous avons fixé pour la présente section, le taux de bonne détection de l'*allégeance esm* (singleton) à 80%. Nous avons également un changement d'allégeance de θ_1 à θ_3 à l'itération 50. (c.-à-d. de ami à ennemi)

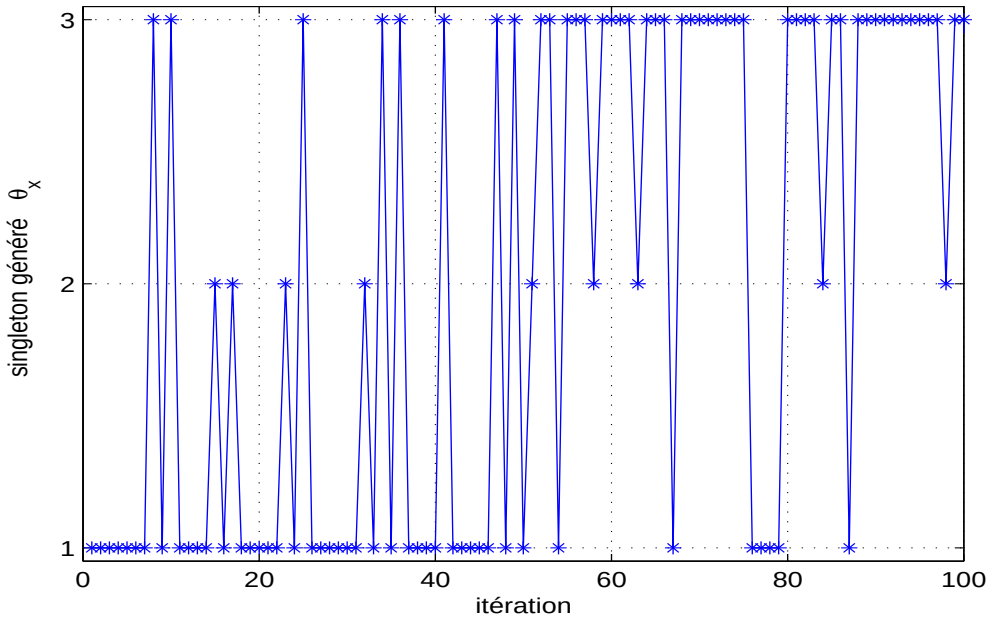


FIG. 5.1 – Identification de l'allégeance selon le senseur ESM

5.2.1 Versions naturelles

Cette section présente les résultats de simulations qui combinent les informations affichées à la figure 5.1 à l'aide de diverses règles de combinaison vues au chapitre 2. Par 'versions naturelles', on veut dire que ce sont les versions originales inaltérées des règles de combinaisons qui sont utilisées.

La figure 5.2 présente les résultats de la combinaison avec la règle du vote majoritaire avec une fenêtre de 5 unités de temps. La figure montre la difficulté qu'a cette règle de résister à une seule, ou une série de mauvaises détections ou de contre-mesures. (c.-à-d. une mauvaise *allégeance esm* qui est attribuée) (voir itérations 75 à 85.) La courbe de θ_2 se maintient, après une période de stabilisation, autour de 0.15, ce qui correspond à son taux de génération.

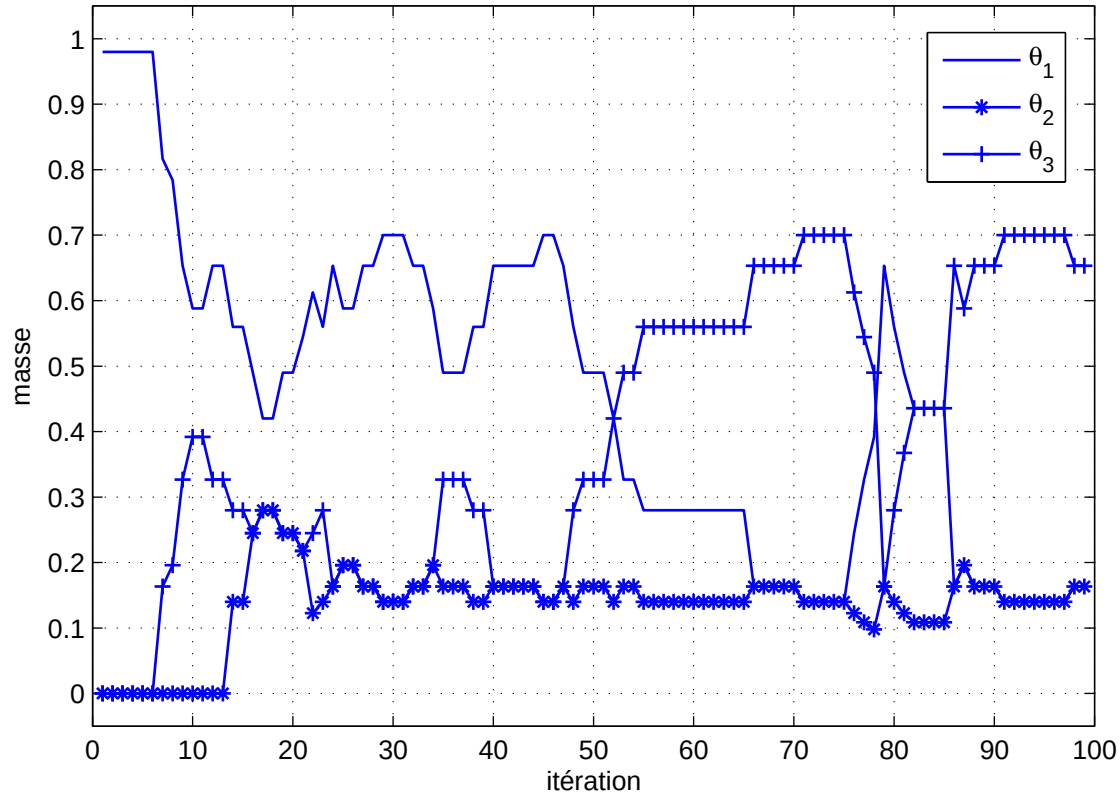


FIG. 5.2 – Résultat de combinaison par la règle du vote majoritaire (Fonction de masse des éléments focaux)

La figure 5.3 présente les résultats pour les règles conjonctive et disjonctive. On y voit clairement leur inhabileté à fonctionner, tel que prévu à la section 4.5.1. Notez que nous

avons négligé l'affichage des courbes de θ_2 (neutre), θ_3 (ennemi), $\theta_1 \cap \theta_2$ (présumé ami), $\theta_2 \cap \theta_3$ (suspect) sur la figure 5.3.1, car leur masse était simplement non significative.

La figure 5.4 montre les résultats lors de l'utilisation de la règle de Dempster. La figure 5.4.1 montre que la règle de Dempster est en mesure de clairement identifier la bonne *allégeance esm* et supporte aisément le changement d'allégeance. Elle est également en mesure de récupérer rapidement après une succession de contre-mesures (voir les itérations 76 à 79). La répartition des probabilités pignistiques pour la règle de Dempster est similaire à la répartition de ses masses. Entre les figures 5.4.1 et 5.4.2, on ne remarque qu'une légère différence en début de courbe pour θ_2 spécifiquement. La similitude des figures s'explique par le fait que cette règle associe des masses non nulles qu'aux singletons. La différence quant à elle, elle s'explique par le fait qu'en début de figure la part de l'ignorance totale est plus importante en proportion aux autres objets.

La DSTd à la figure 5.5 est affecté du même problème d'accumulation que celle à l'ignorance totale pour la règle disjonctive.

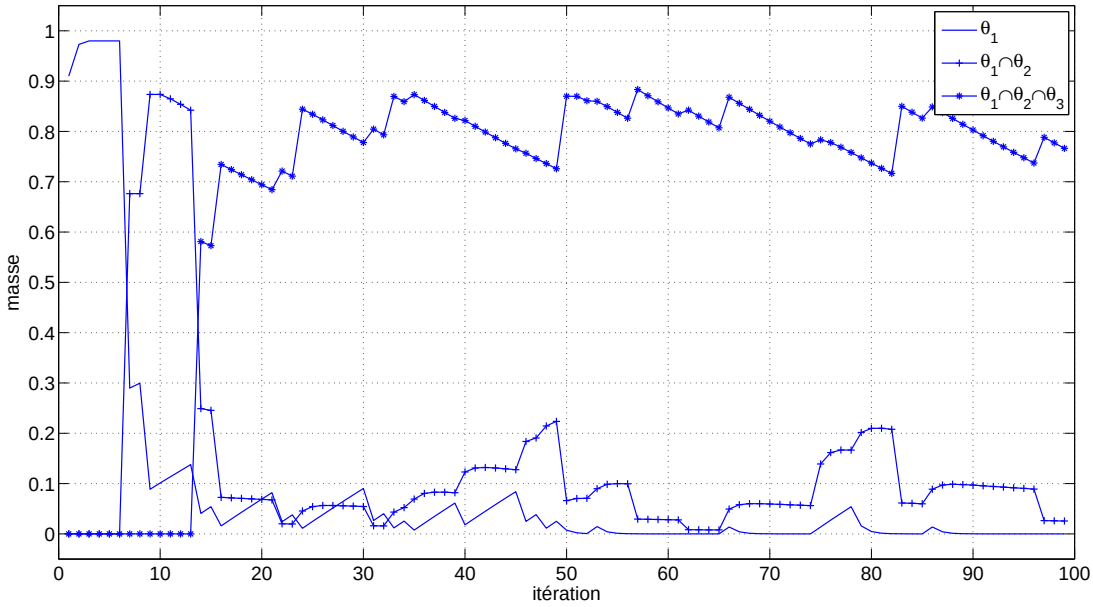
La figure 5.6 montre le résultat pour la règle SACR. On relève une réponse fortement oscillatoire des masses attribuées au bon singleton tel que discuté à la section 4.6. Ce problème rend la règle inappropriée pour la prise de décision, du moins, à partir de la masse. Toutefois, des solutions montrées à la section 4.6.1 sont possibles.

Outre l'oscillation, la SACR donne correctement une masse majoritaire au bon singleton lors d'absence de contre-mesures. Lors de contre-mesures, elle réagit fortement et ne dénote aucune résistance d'où le problème d'oscillation.

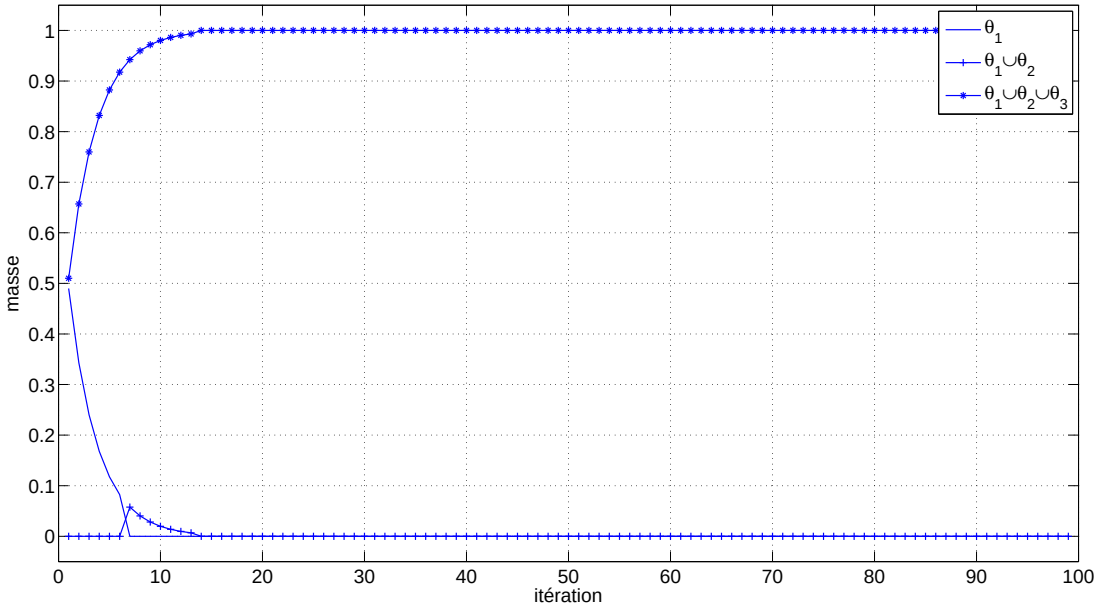
La EACR présentée à la figure 5.7, répartit une bonne part de la masse à des intersections et des unions. Cela affaiblit de façon importante sa capacité à prendre des décisions directement sur des singletons à partir de la fonction de masse. À cela s'ajoute le fort niveau d'oscillation dû à son incapacité à résister aux contre-mesures. Le choix de cette règle de combinaison est remis en question dans la version initiale pour notre cas de simulation.

Pour résoudre ce problème, on avait pensé utiliser, au lieu des masses des singletons et des intersections distinctement, d'ajouter la masse des intersections avec A au singleton A lui-même. La question est de choisir à quel singleton, et dans quelle proportion la masse de l'objet $A \cap B$ doit être attribuée. Ceci est un obstacle important. Notez également la similarité de cette idée et du problème avec la TPG.

On remarque tout de même que si l'on considère θ_1 (ami) avant le changement

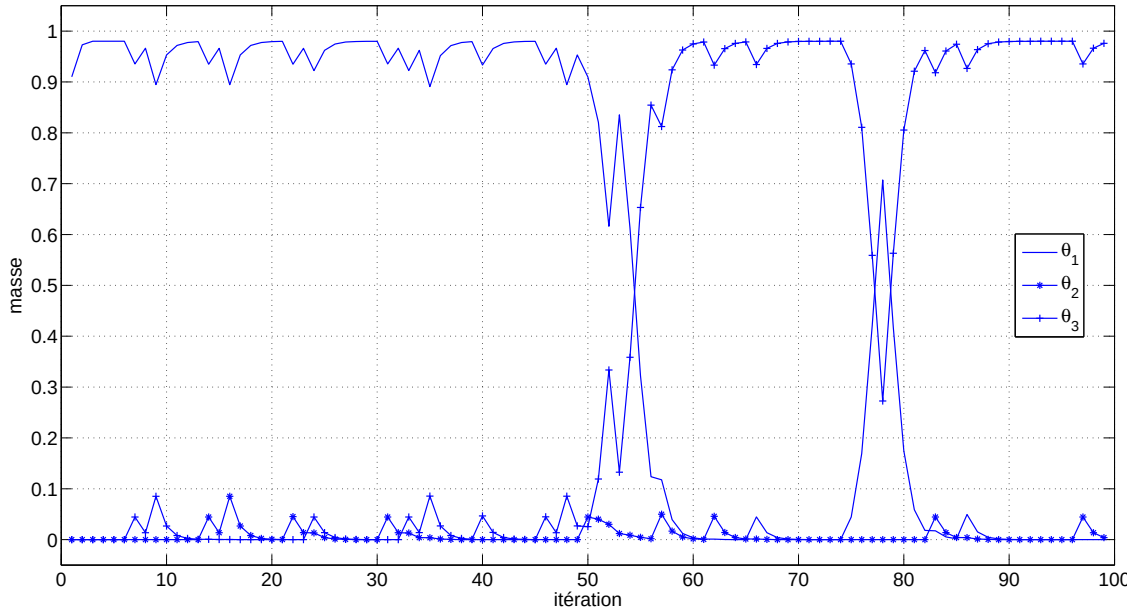


5.3.1: Résultat de combinaison par la règle conjonctive

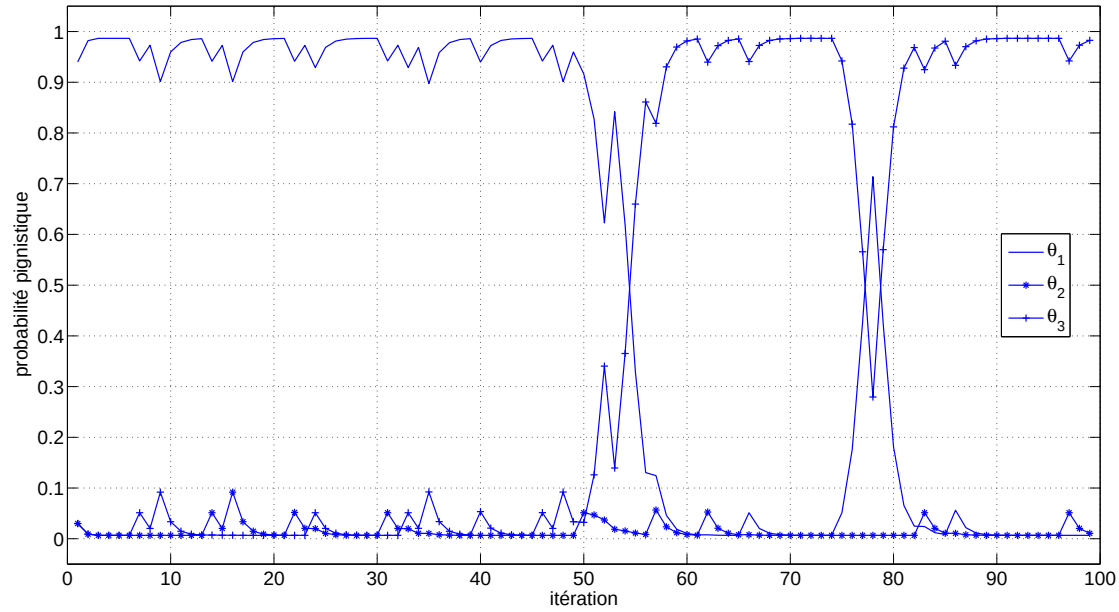


5.3.2: Résultat de combinaison par la règle disjonctive

FIG. 5.3 – Résultat de combinaison par des règles de base dans D^{Θ}



5.4.1: Masse des singletons



5.4.2: Probabilité pignistique

FIG. 5.4 – Résultat de combinaison par la règle de Dempster

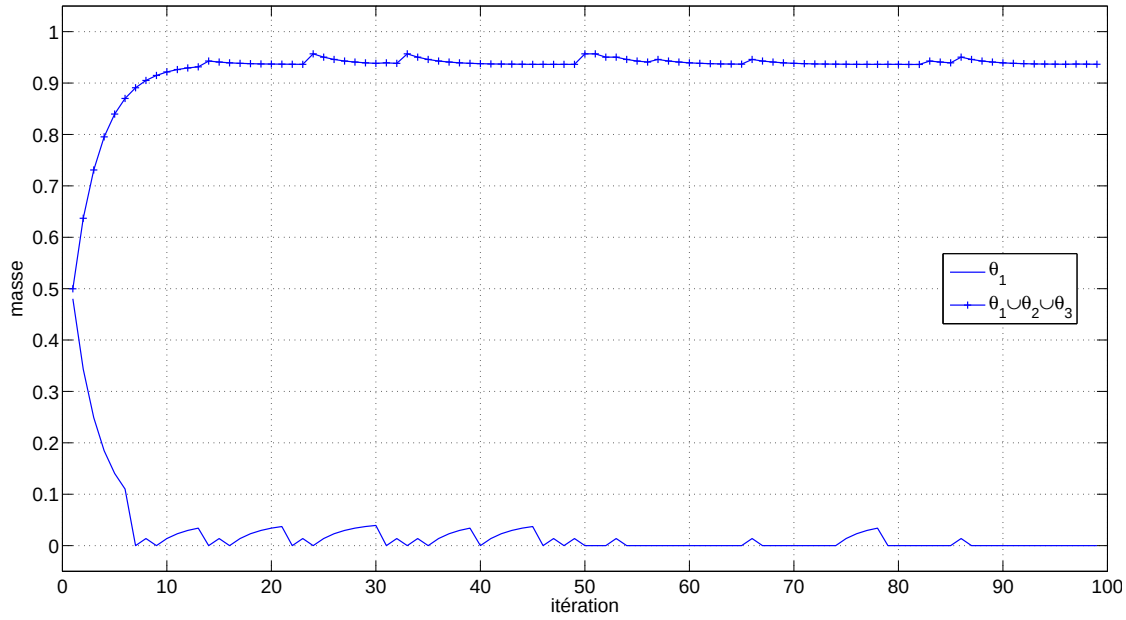


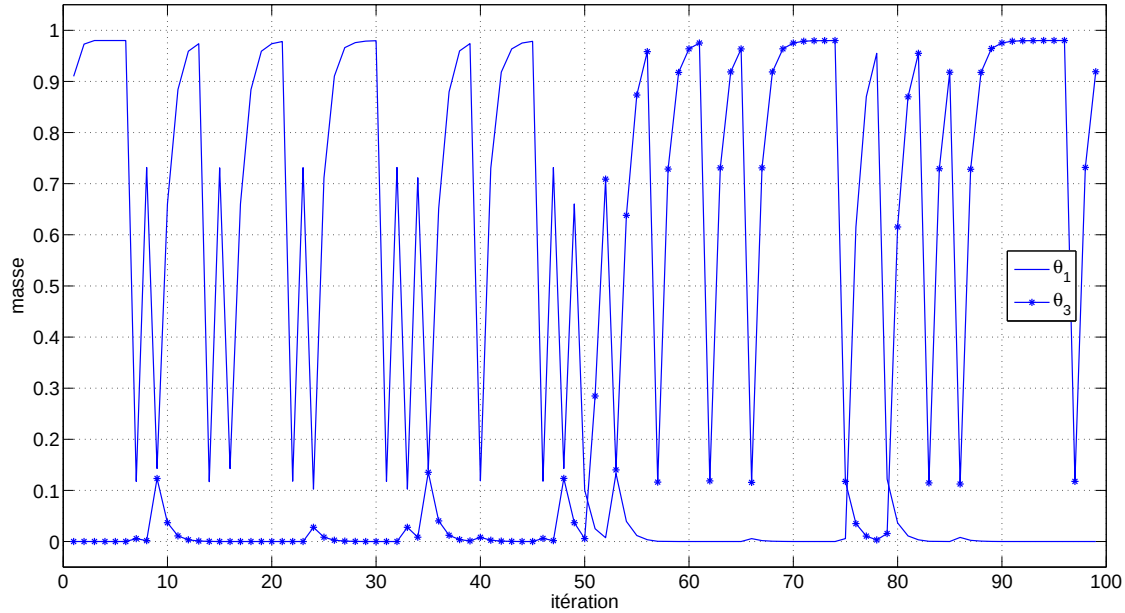
FIG. 5.5 – Résultat de combinaison par la règle de DS disjonctive (DSTd)

d'allégeance, avec les intersections et unions qui impliquent θ_1, leur masse est majoritaire comme il se doit. Il en est de même pour θ_3 (ennemi) après le changement d'allégeance.

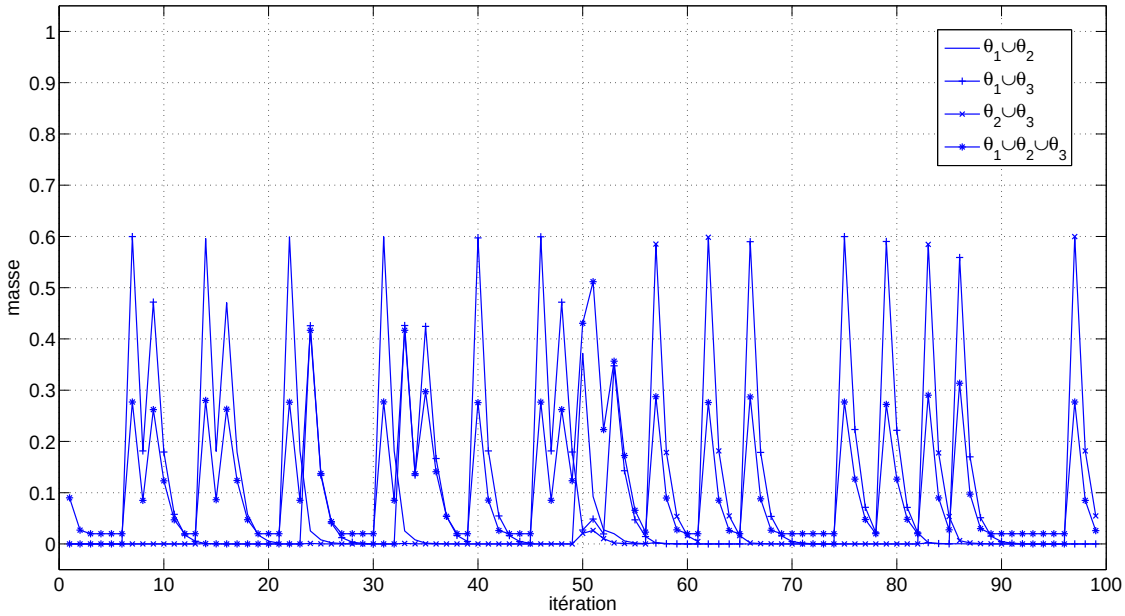
La PCR présentée à la figure 5.8, donne une majorité des masses à θ_1 ou des intersections avec θ_1 avant le changement d'allégeance, tel que prévu. Il en est de même pour θ_3 après le changement d'allégeance.

La DSmH a, à la figure 5.9, un comportement similaire à la PCR et la EACR. Toutefois, la DSmH ne conserve qu'une masse minime aux singletons car une grande partie de la masse est envoyée aux intersections ou aux unions selon ce qui est observé à la figure 5.9.2. Malgré cela, elle répond bien au changement d'allégeance.

Comme on le constate à la figure 5.10, la DSmH avec décision à partir de la probabilité pignistique généralisée n'échappe pas au phénomène rencontré précédemment. Il semble même être plus intense ici. Outre ce problème, la DSmH avec TPG semble bien réagir au changement d'allégeance et bien résister aux contre-mesures. On note également une présence importante de la probabilité pignistique accordée aux intersections (voir figure 5.10.2). Cela est tout à fait normal et demeure en accord avec les propriétés de la TPG et de la théorie des probabilités. Notez également qu'il est normal dans le cas de probabilités pignistiques que l'on ait des valeurs près de l'unité pour plusieurs singletons à la fois. Il faut comprendre que les probabilités pignistiques suivent la règle de Bayes et que $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$ par exemple.

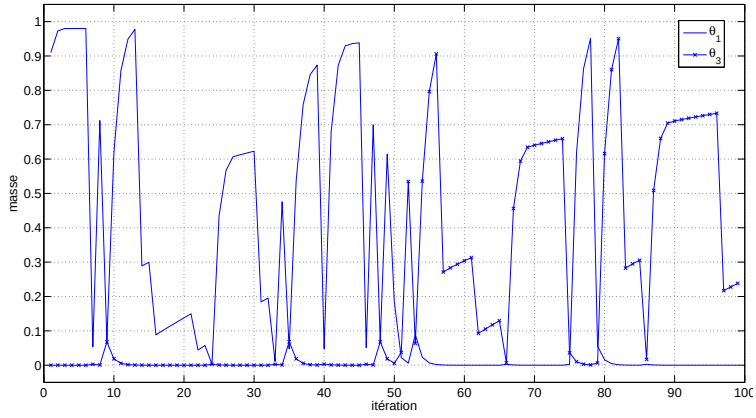


5.6.1: Masse des singletons θ_1 et θ_3

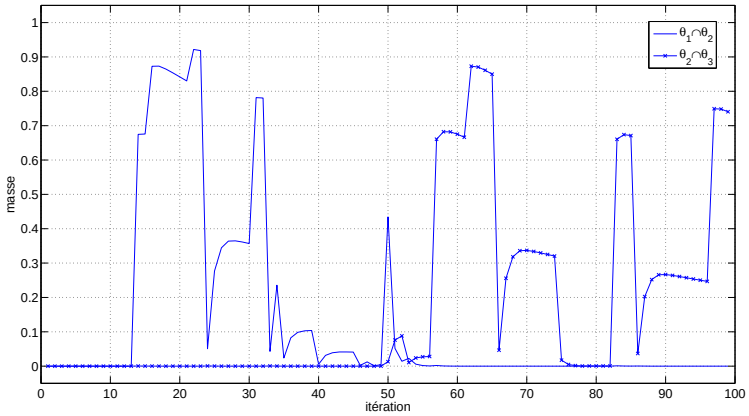


5.6.2: Masse des ignorances partielles et totale

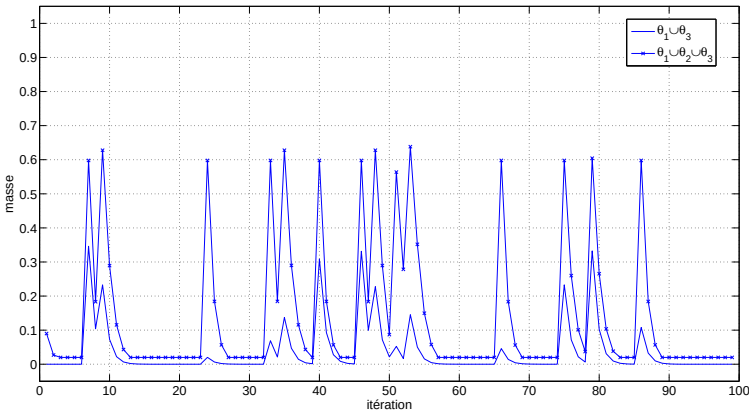
FIG. 5.6 – Résultat de combinaison par la règle SACR



5.7.1: Masse des singletons θ_1, θ_3

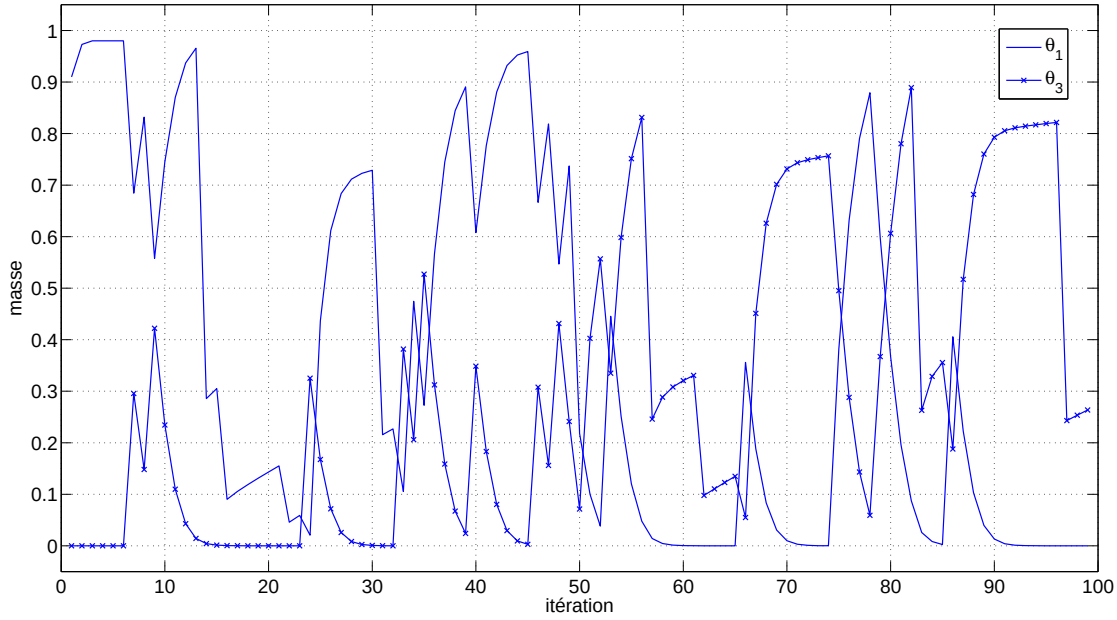


5.7.2: Masse de $\theta_1 \cap \theta_2$, $\theta_2 \cap \theta_3$

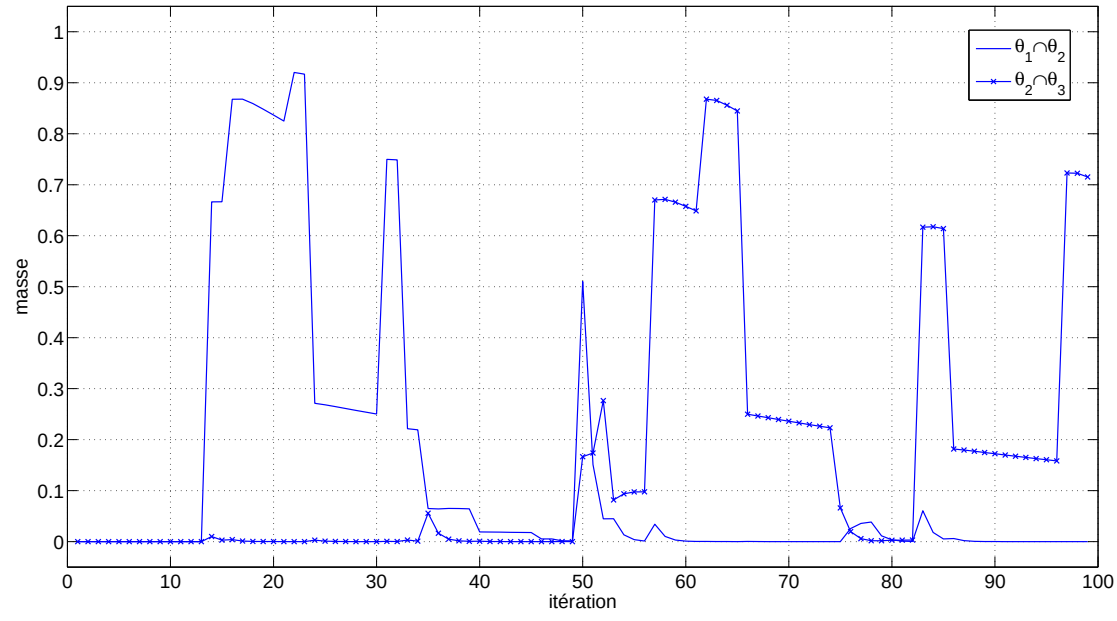


5.7.3: Masse de $\theta_1 \cup \theta_2$, $\theta_1 \cup \theta_2 \cup \theta_3$

FIG. 5.7 – Résultat de combinaison par la règle EACR

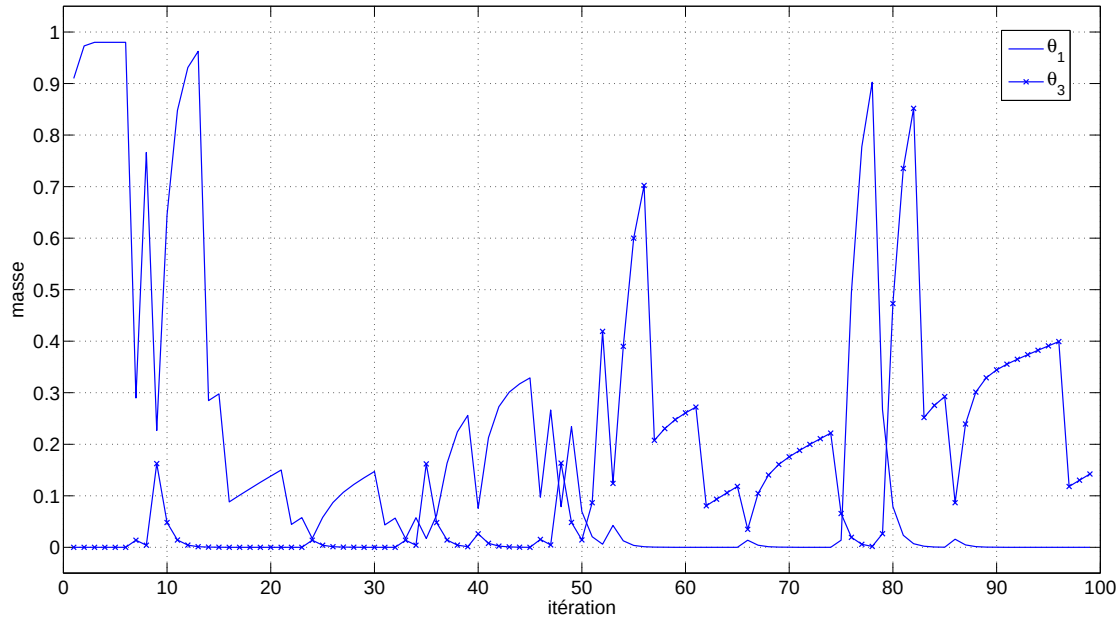


5.8.1: Masse des singletons θ_1, θ_3

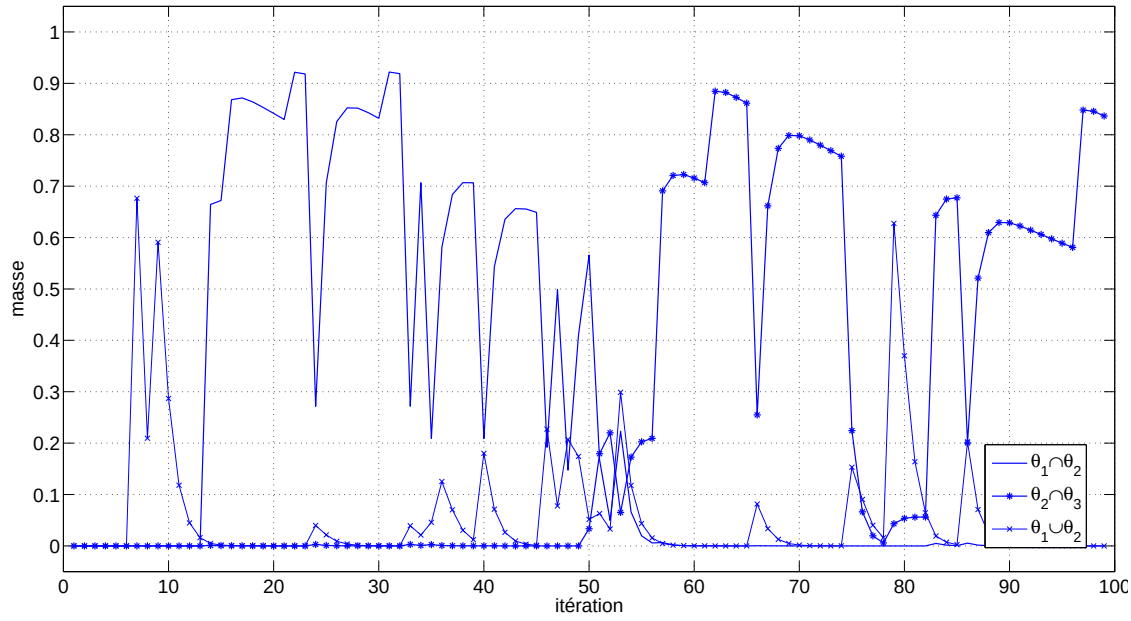


5.8.2: Masse de $\theta_1 \cap \theta_2$, $\theta_2 \cap \theta_3$

FIG. 5.8 – Résultat de combinaison par la règle PCR

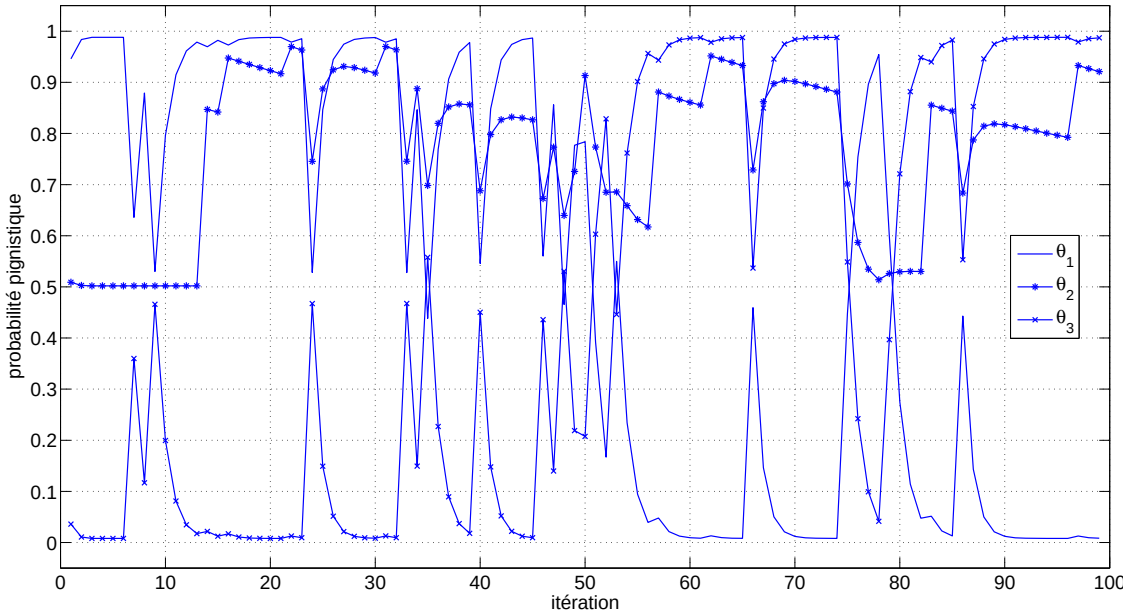


5.9.1: Masse des singletons θ_1, θ_3

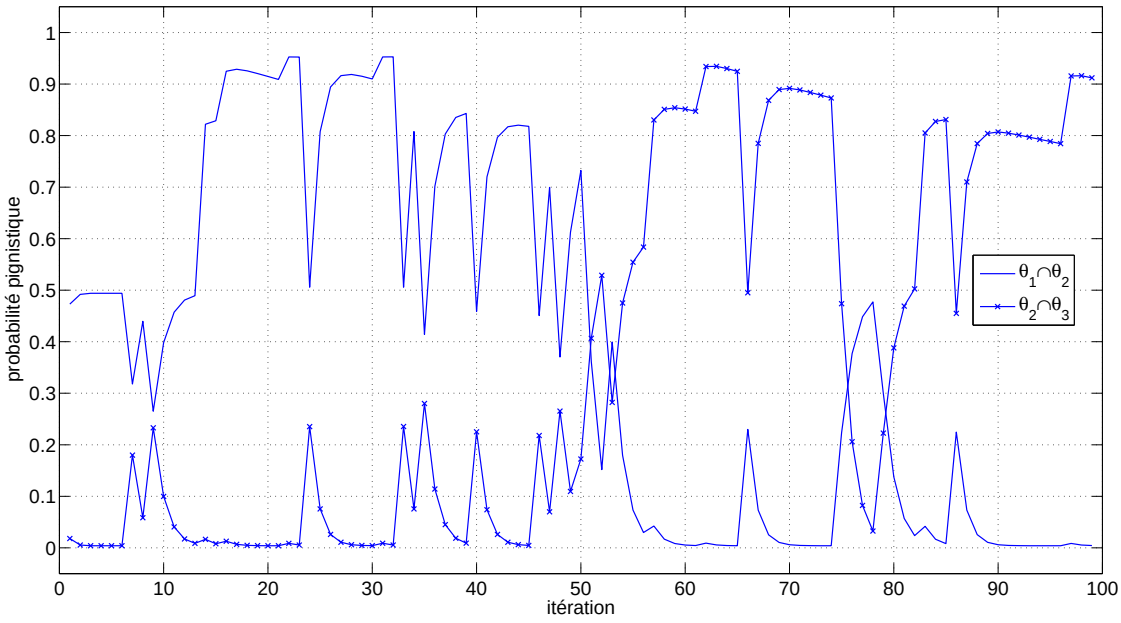


5.9.2: Masse de $\theta_1 \cap \theta_2$, $\theta_2 \cap \theta_3$, $\theta_1 \cup \theta_3$

FIG. 5.9 – Résultat de combinaison par la règle DSmH



5.10.1: Probabilité pignistique des singletons θ_1, θ_2 et θ_3



5.10.2: Probabilité pignistique de $\theta_1 \cap \theta_2$ et $\theta_2 \cap \theta_3$

FIG. 5.10 – Résultat de combinaison par la règle DSmH suite à une TPG

5.2.2 Versions quasi-associatives

Les versions quasi-associatives de nos simulations sont celles qui intègrent la quasi-associativité telle que présentée à la section 4.3.1 par défaut. La version basée sur la règle de DS telle que présentée à la section 4.5.4 est spécifiée dans le cas contraire.

À la figure 5.11, le cas quasi-associatif de la PCR répartit une grande part de la masse à l'ignorance totale. Ce qui laisse peu de place aux ensembles possibles. Il est également touché par une grande sensibilité aux contre-mesures.

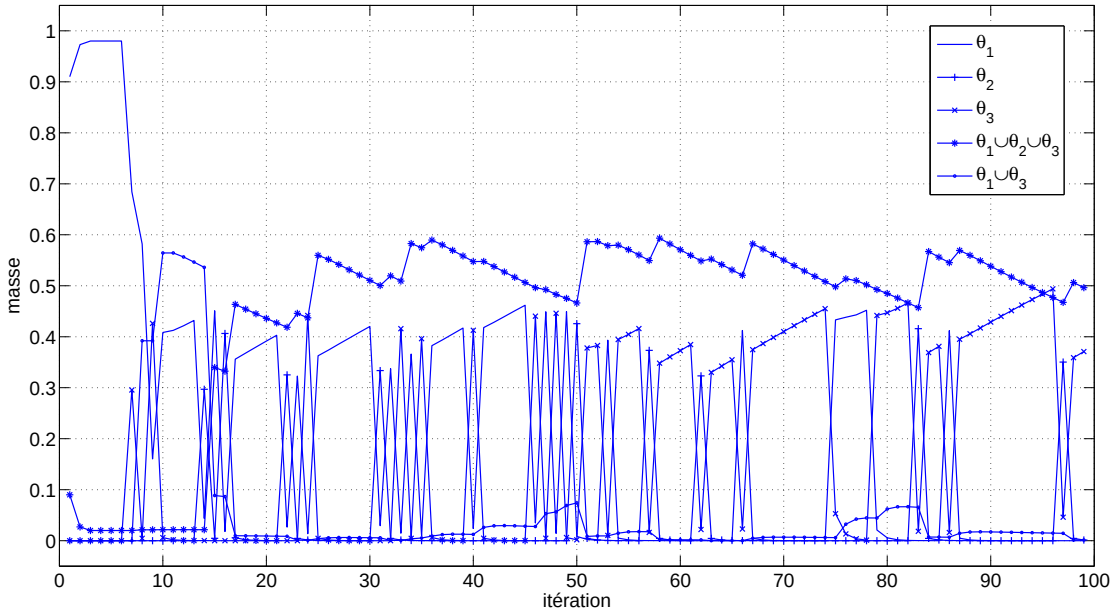


FIG. 5.11 – Résultat de combinaison par la règle PCR avec quasi-associativité

La figure 5.12 montre les résultats de combinaison de la PCR qui utilise la quasi-associativité basée sur la règle de Dempster. Cette fois, on est en mesure, malgré l'oscillation, de clairement identifier le bon singleton au bon moment. La réactance au changement d'allégeance est bonne, mais l'intensité des oscillations montre une difficulté à gérer les contre-mesures. Il y a toutefois récupération très rapide suite à ces contre-mesures.

L'équivalent de la figure 5.12 avec filtre sur les sorties est présenté à la figure 5.13. On obtient donc une PCR quasi-associative basée sur la règle de Dempster avec filtre. Tel que prévu, les oscillations sont significativement plus faibles. Toutefois, le comportement général des courbes est identique. On obtient une bien meilleure résistance au changement d'allégeance.

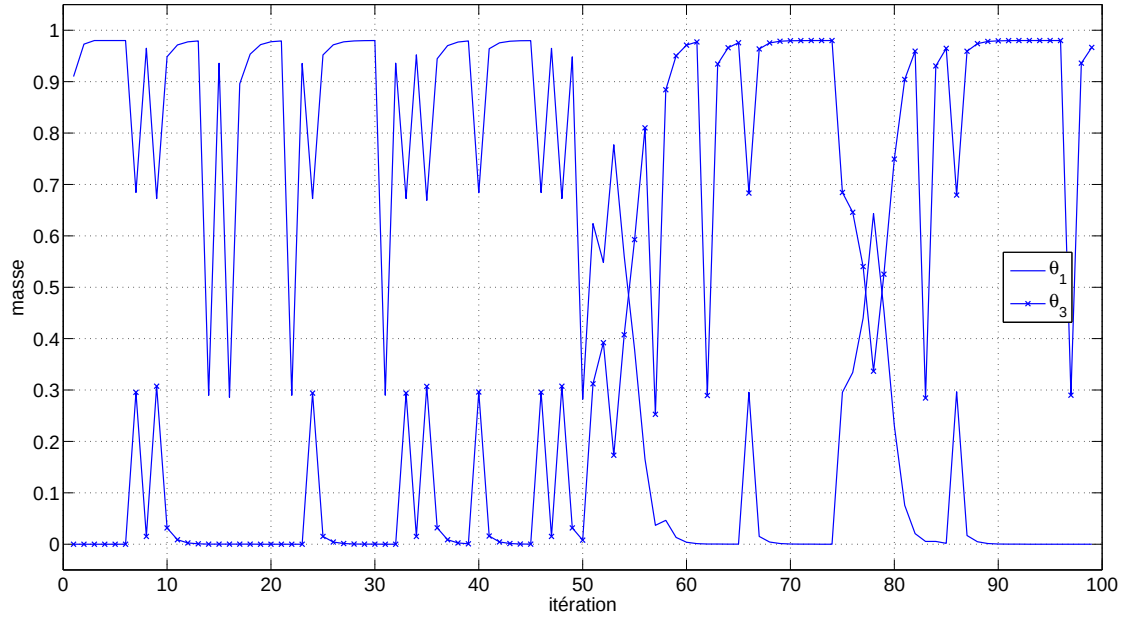


FIG. 5.12 – Résultat de combinaison par la règle PCR avec quasi-associativité basée sur la Dempster

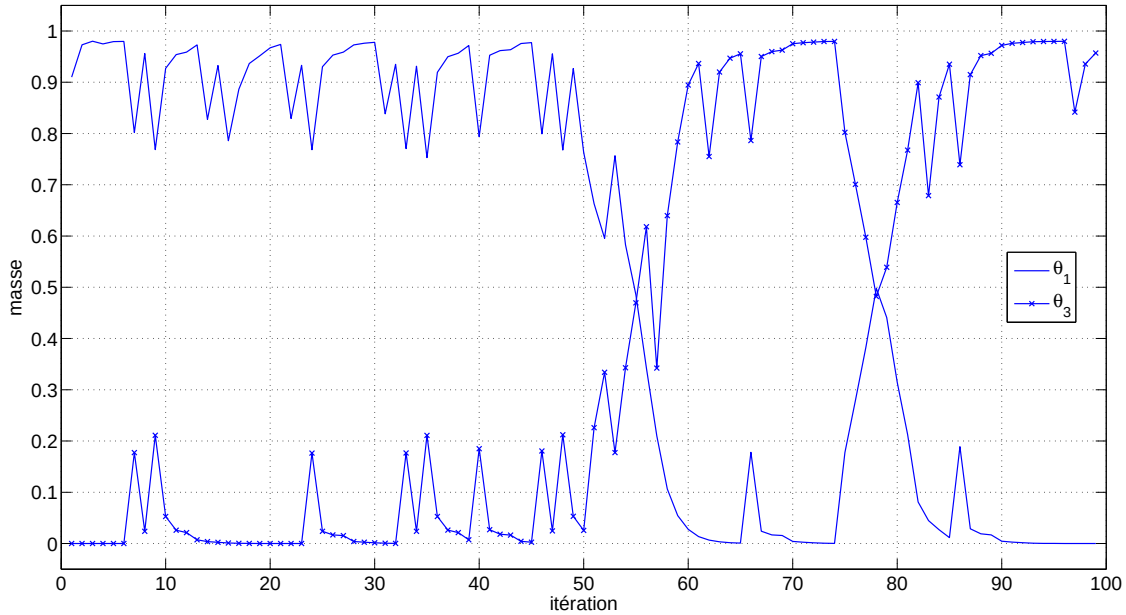


FIG. 5.13 – Résultat de combinaison par la règle PCR avec quasi-associativité basée sur la Dempster avec filtre sur les sorties

La version de quasi-associativité, basée sur la DST, de la DSmH à sorties filtrées est présentée à la figure 5.14. Cette dernière s'apparente beaucoup plus à la PCR aux mêmes adaptations (voir figure 5.13) obtenant une forte masse aux singletons et une oscillation minime dénotant une très bonne résistance aux contre-mesures. On remarque également une bonne réactance au changement d'allégeance.

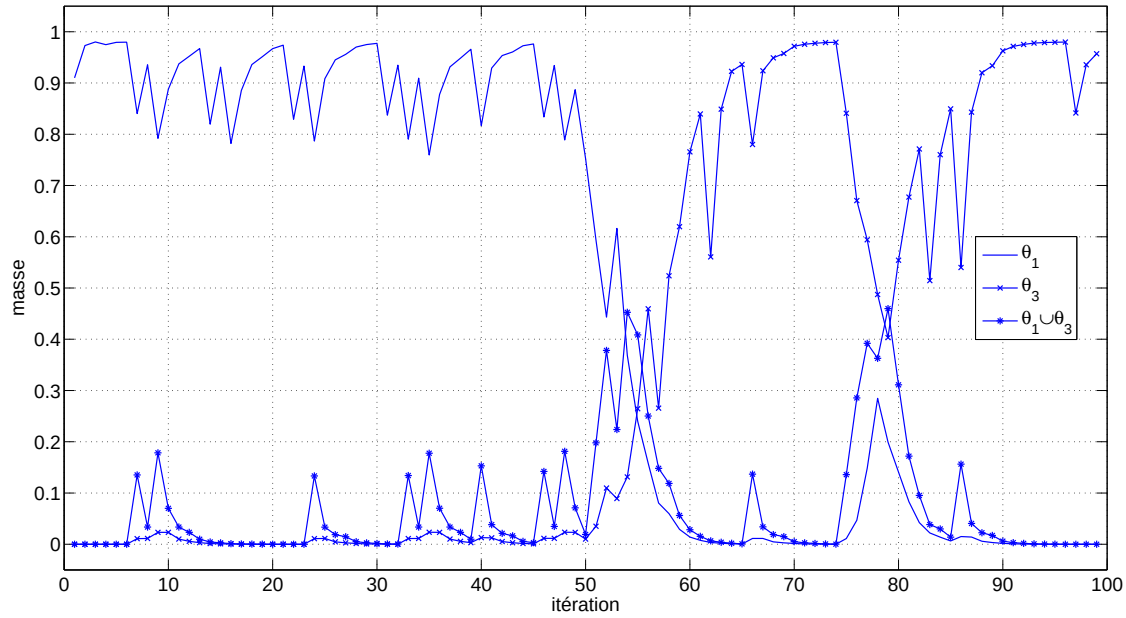


FIG. 5.14 – Résultat de combinaison par la règle DSmH avec quasi-associativité basée sur la Dempster avec filtre sur les sorties

5.3 Résultats et analyse de simulations Monte-Carlo

Avant de procéder, on rappelle la distinction entre la définition du taux de bonnes décisions que l'on a utilisée et celle qui prévaut en pratique. Tel qu'expliqué à la section 3.2.6, en pratique, une cible identifiée comme étant amie, présumée amie, et neutre, ne sera pas l'objet d'attaque. La décision dans ce cas sera donc de ne pas procéder agressivement, et ce sera le bon choix. Ainsi, selon différents contextes, différentes décisions peuvent être considérées comme bonnes ou mauvaises.

Toutefois, dans ce mémoire, on définit la bonne décision comme étant l'identification avec succès du singleton représentant l'*allégeance esm* (θ_1 =amie, θ_2 =neutre, θ_3 =ennemie). Donc, l'ensemble ou singleton qui a la probabilité pignistique supérieure est l'ensemble de l'allégeance présumée de la cible. La décision est ainsi considérée comme bonne si l'allégeance avec la probabilité pignistique supérieure est bien celle qui représente la cible à ce moment là.

Notez que ce chapitre ne présente que les résultats principaux qui se démarquent. L'annexe A présente différents cas supplémentaires qui ont permis de tirer diverses conclusions sur le comportement des combinaisons sous différents contextes et paramètres testés.

5.3.1 Décision avec les probabilités pignistiques

La figure 5.15 fait la représentation graphique des allégeances possibles sous le STANAG 1241 avec les correspondances aux singletons est là pour rappeler qu'à la base, sous la théorie de l'évidence, un singleton θ_1 par exemple, inclut la zone couverte par l'*allégeance stanag* 'amie' ainsi que 'présumée amie'. Dans le cas du singleton θ_3 par exemple, ce dernier inclut la zone couverte par l'*allégeance stanag* 'ennemie' et 'suspecte'. Ainsi, lors de l'évaluation du taux de bonnes décisions à l'aide des probabilités pignistiques lorsque l'on détermine via une règle de combinaison que l'allégeance d'une cible donne une probabilité pignistique de $\theta_1 = 0.8$, c'est en fait $Pr(\theta_1) = 0.80$ où $Pr(amie) = Pr(\theta_1) - Pr(\theta_1 \cap \theta_2)$, donc $Pr(amie) \neq Pr(\theta_1)$. Avec ce cas précis et la répartition des allégeances (c.-à-d. *allégeance stanag*) (voir figure 5.15) sur laquelle on expérimente, dans les 50 premières itérations, où la cible est en réalité d'*allégeance esm* amie, la bonne décision correspond à un comportement amical de notre part, ainsi, à θ_1 qui inclut 'amie' et 'présumée amie'. Il s'en suit que dans les 50 dernières itérations, où la cible est en réalité d'*allégeance esm* ennemie, la bonne décision correspond à un comportement hostile de notre part, ainsi à θ_3 qui inclut 'ennemie' et 'suspecte'.

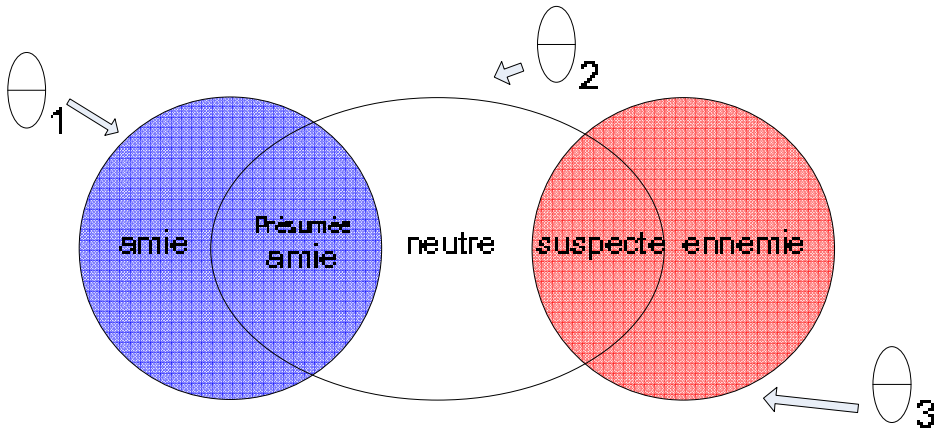


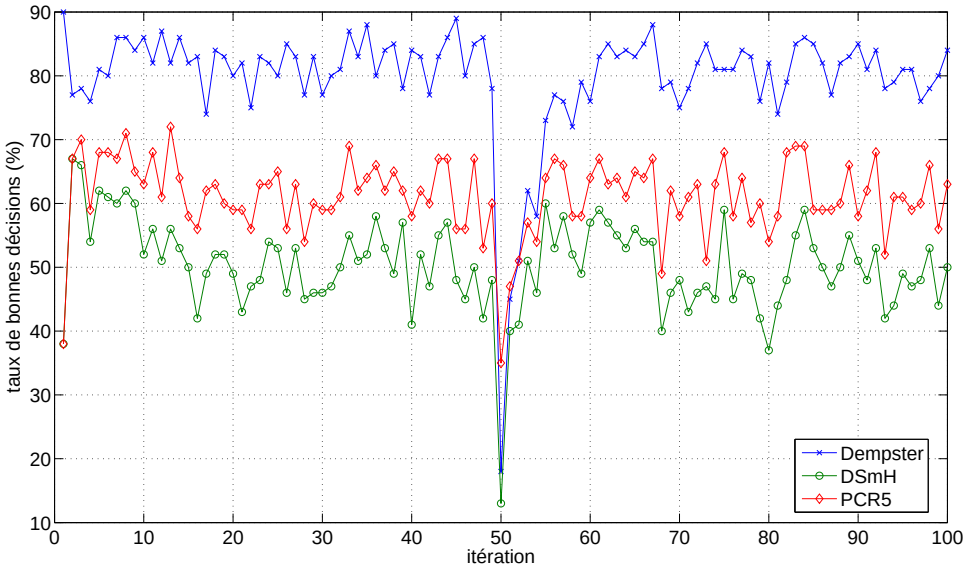
FIG. 5.15 – Représentation graphique des allégeances possibles sous le STANAG 1241 avec correspondances aux singletons

Les résultats des simulations de la présente section permettent de relever les remarques suivantes (figures 5.16 à 5.19) :

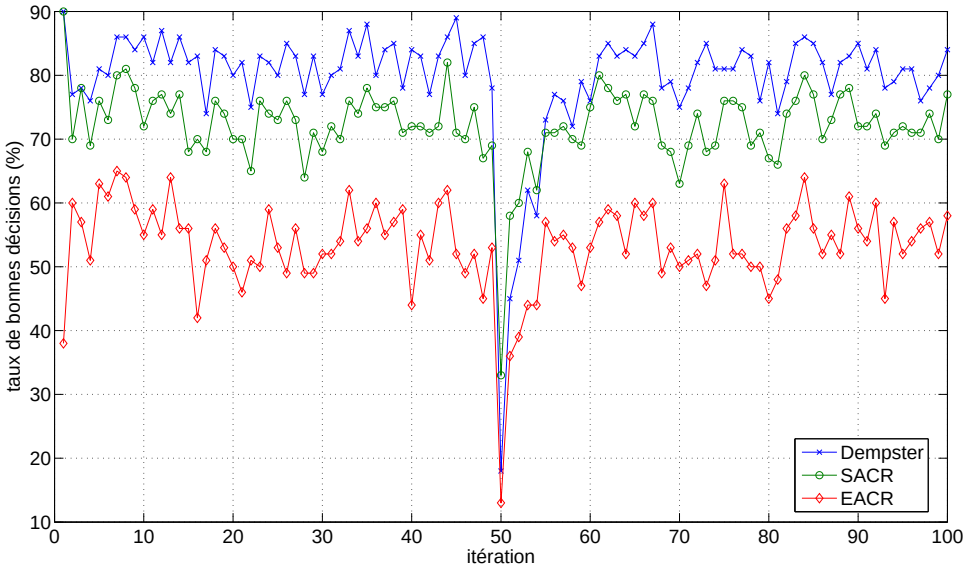
- Le taux de bonnes décisions est légèrement plus élevé, lorsque la masse attribuée au singleton détecté par le senseur ESM est plus élevée.
- L'intervalle de confiance est légèrement plus large lorsque la masse attribuée au singleton détecté par le senseur ESM est plus élevée.
- Les résultats de toutes les règles s'améliorent significativement lorsque le taux de bonnes détections de l'*allégeance esm* s'accroît. Dans les prochaines simulations ce taux de détections augmente de un pourcent par itération, trente itérations après un changement d'allégeance.
- Même en tenant compte de l'intervalle de confiance à 95%, lorsque l'on compare un cas à masse identique attribuée au singleton détectée par senseur ESM, le cas avec filtre sur les entrées obtient un taux de bonnes décisions supérieur pour la DSmH, PCR5, SACR et EACR. La règle de Dempster plafonne déjà à 100% d'efficacité.
- Les règles DSmH, PCR5, SACR et EACR ont un taux de bonnes décisions inférieur d'approximativement 30% à un cas équivalent ayant un taux de bonnes détections d'*allégeance esm* à 60% au lieu de 80%. La DS est affectée mais dans une moindre mesure : elle demeure au-delà de 90% de bonnes décisions sauf durant la période d'initialisation et au changement d'allégeance.
- Le taux de bonnes décisions est légèrement plus élevé lorsque le seuillage de l'ignorance totale est à 2% au lieu de 5%, même en tenant compte de l'intervalle de confiance. Le taux de bonnes décisions pour la règle de DS plafonne à 100%, hormis durant les dix itérations d'initialisation et la douzaine d'itérations après le changement d'allégeance.
- Les règles DSmH, PCR5, SACR et EACR ont toutes réactance plus élevée au

changement d'allégeance par rapport à la règle de combinaison de Dempster-Shafer.

Notez que la décision avec les probabilités pignistiques se fait par la sélection de l'ensemble ayant la probabilité pignistique la plus élevée comme étant l'ensemble de l'allégeance présumée de la cible.

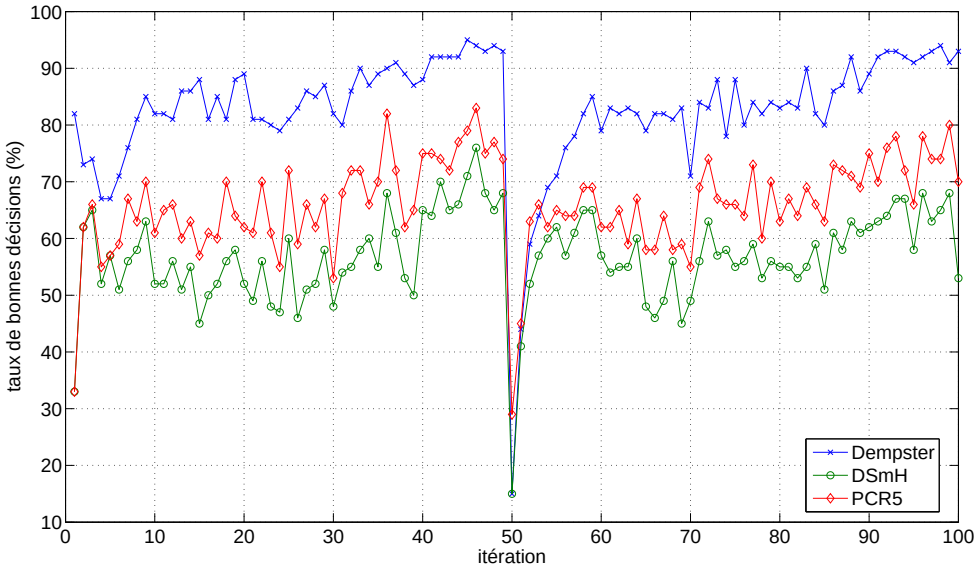


5.16.1: Règles de Dempster, DSmH, et PCR5

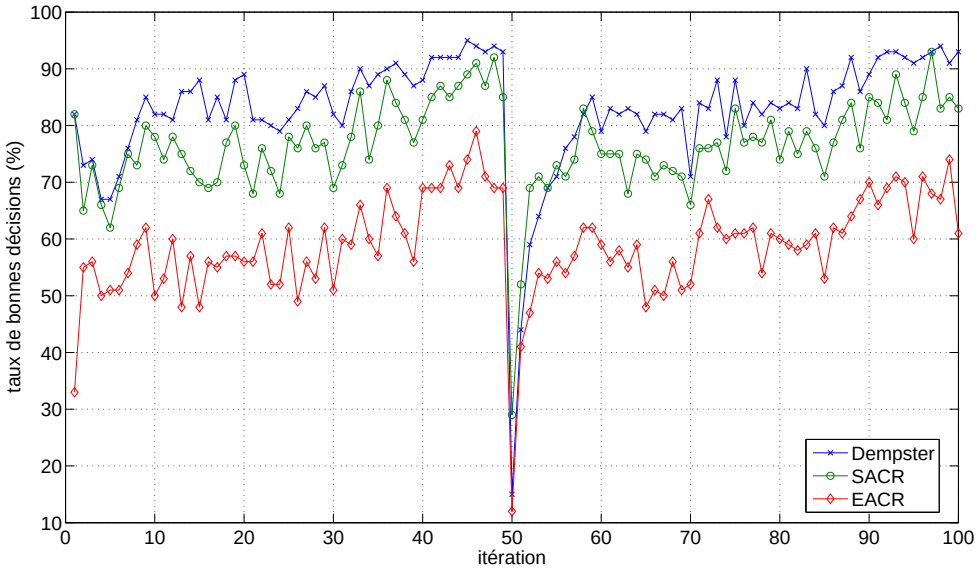


5.16.2: Règles de Dempster, SACR, et EACR

FIG. 5.16 – Taux de bonnes décisions basé sur la probabilité pignistique avec une masse de 90% attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%

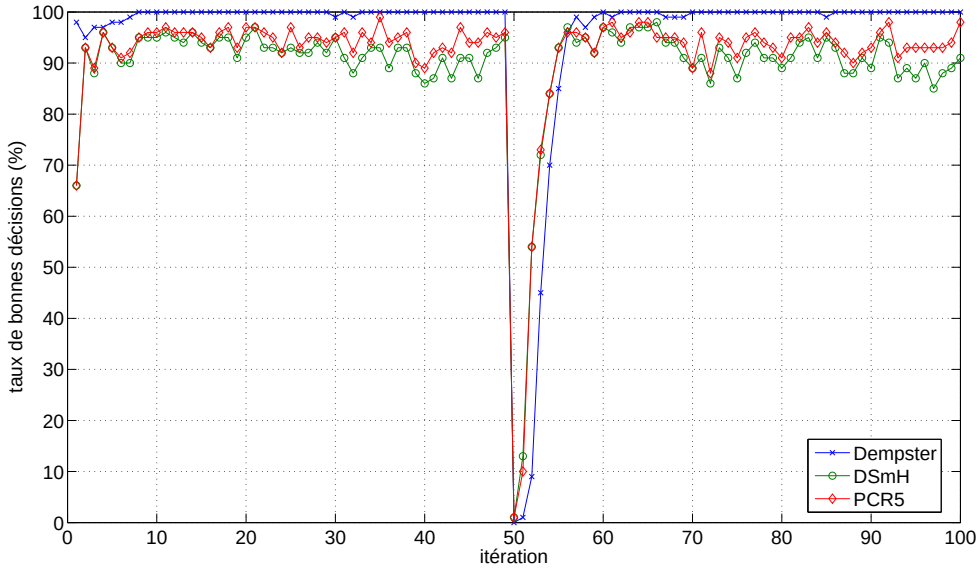


5.17.1: Règles de Dempster, DSmH, et PCR5

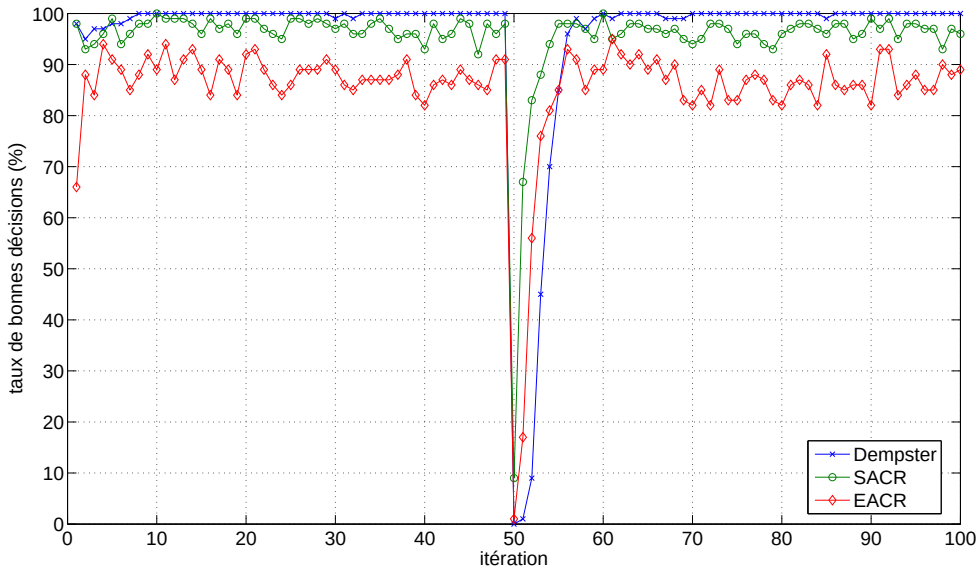


5.17.2: Règles de Dempster, SACR, et EACR

FIG. 5.17 – Taux de bonnes décisions basé sur la probabilité pignistique avec une masse de 90% attribuée à l'*allégeance esm* détectée, un taux de détection de 60% avec croissance cummulative d'un pourcent par itération après 30 itérations sans changement d'allégeance et un seuillage de I_t à 2%

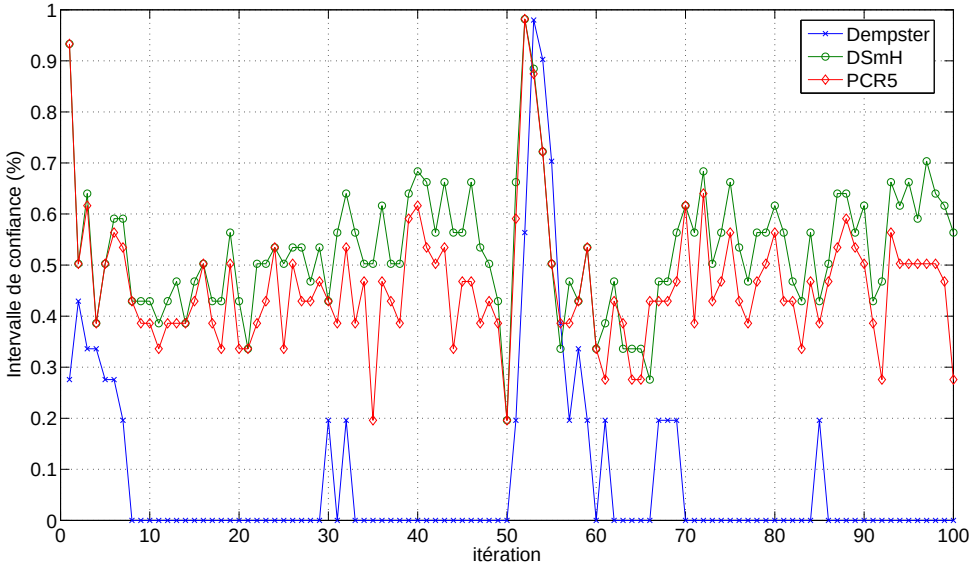


5.18.1: Règles de Dempster, DSmH, et PCR5

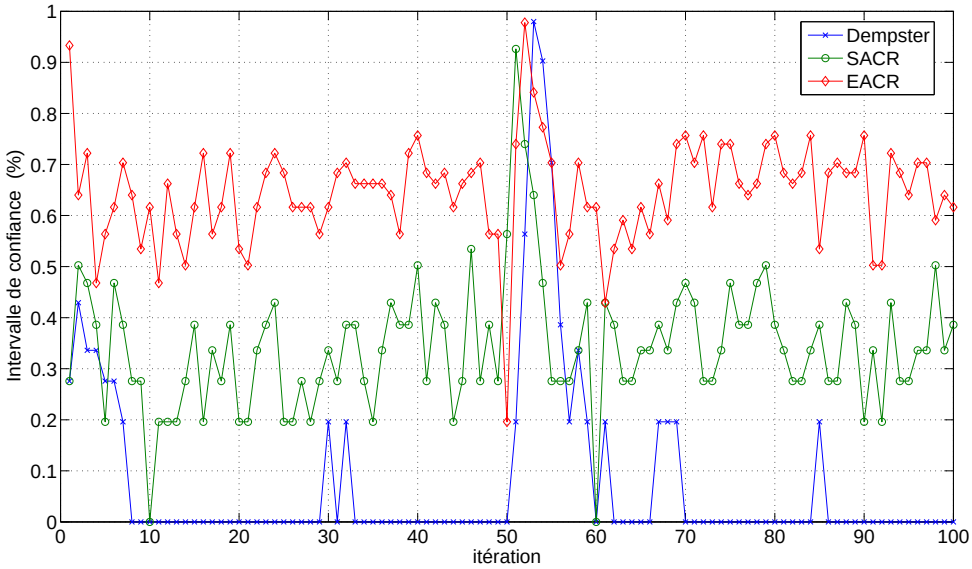


5.18.2: Règles de Dempster, SACR, et EACR

FIG. 5.18 – Taux de bonnes décisions basé sur la probabilité pignistique avec un filtre sur les valeurs de masses attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 5%



5.19.1: Intervalle de confiance pour les règles de Dempster, DSmH, et PCR5



5.19.2: Intervalle de confiance pour les règles de Dempster, SACR, et EACR

FIG. 5.19 – Intervalle de confiance de la simulation de la figure 5.18

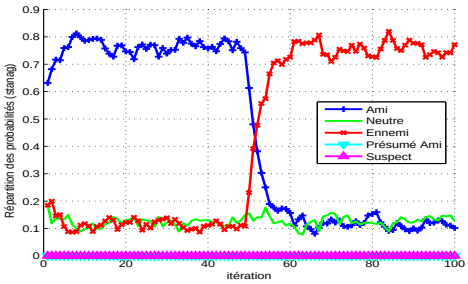
5.3.2 Décision avec les probabilités réparties suivant la classification du Stanag 1241

Cette section présente les résultats de simulations Monte-Carlo avec les probabilités pignistiques (Que l'on désignera : probabilités (stanag).) suivant la répartition des ensembles présentée à la figure 3.1, prescrite par la classification du Stanag 1241. Ces probabilités (stanag) fonctionnent sur les ensembles représentant les *allégeances stanag* et suivent la répartition des ensembles comme décrite à la section 3.1.3. La figure 3.3 montre cette répartition des ensembles. Les résultats des simulations sont obtenus d'après les fonctions de masse converties en probabilités pignistiques via la transformation pignistique généralisée, ou classique selon le cadre de raisonnement.

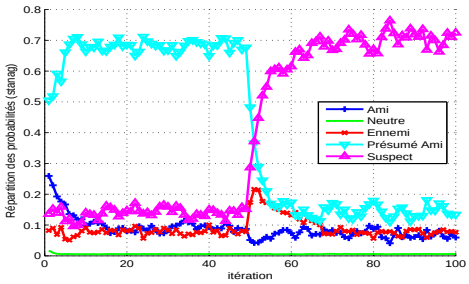
La première partie présente la répartition des probabilités (stanag) pour le cas où une masse de 90% est attribuée à l'*allégeance esm* détectée, un taux de détection de l'*allégeance esm* à 60% et un seuillage de I_t à 2% à la figure 5.20. Le cas où un filtre sur les entrées est appliqué, un taux de détection de l'*allégeance esm* de 80% et un seuillage de I_t à 5% est ensuite montré à la figure 5.21.

Les résultats des simulations de la présente section (figures 5.20 à 5.23) permettent de relever les points suivantes :

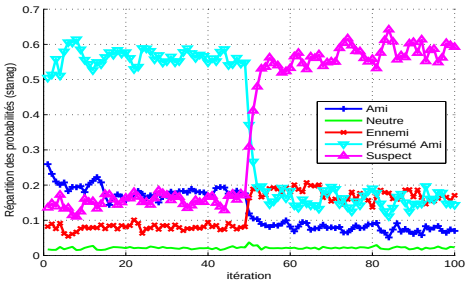
- Il y a amélioration de la stabilité des décisions lorsqu'il y a augmentation du taux de détection de l'*allégeance esm*. Aussi, le taux de bonnes décisions s'améliore ou demeure stable.
- Il y a détérioration de la stabilité des décisions lorsque la masse à l'*allégeance esm* détectée augmente. Cela est dû à l'augmentation du niveau de conflit.
- Il y a peu ou pas d'amélioration lorsque le filtre de seuillage de l'ignorance totale passe de 2 à 5%.
- Il y a amélioration, ainsi qu'une meilleure stabilité des décisions lorsque le filtre sur les entrées est actif. Ce dernier atténue le niveau de conflit.
- Les règles qui poussent la masse, et ainsi la probabilité pignistique, rapidement vers les intersections ont plus de difficulté à obtenir un niveau élevé de probabilité (stanag) sur le singleton identifiant l'allégeance de la cible. Ces règles sont toutefois en mesure d'être plus spécifique et d'utiliser la répartition des *allégeance stanag*. Cela n'est pas possible pour les règles de Dempster et SACR qui fonctionnent sous 2^Θ car ces règles ne sont pas en mesure d'envoyer une masse, et ainsi une probabilité pignistique, vers les intersections. Cela semblerait donner un avantage aux règles sous 2^Θ, mais aurait le désavantage de ne pas utiliser le STANAG 1241.



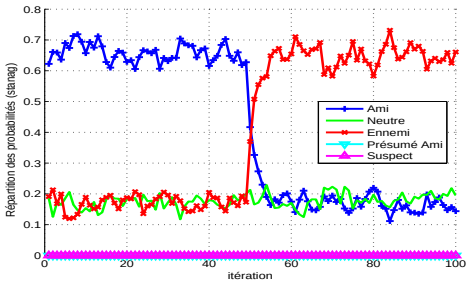
5.20.1: Résultats de la règle de Dempster



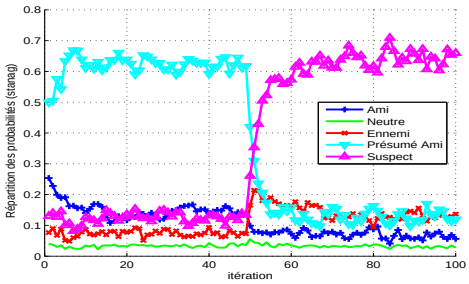
5.20.2: Résultats de la règle de DSmH



5.20.3: Résultats de la règle PCR5

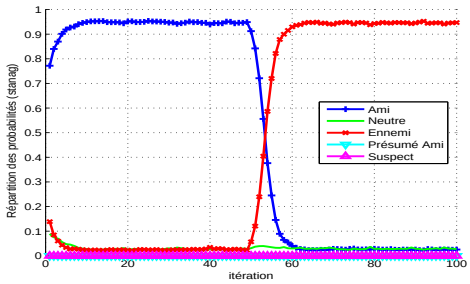


5.20.4: Résultats de la règle SACR

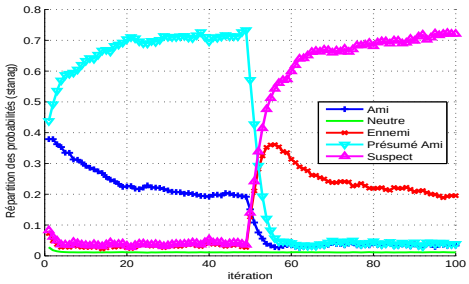


5.20.5: Résultats de la règle EACR

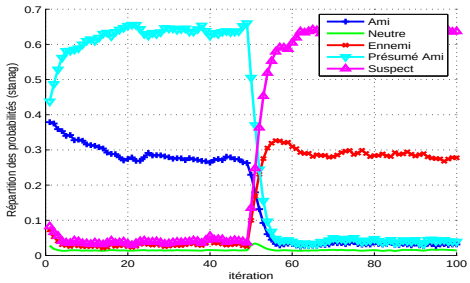
FIG. 5.20 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec une masse de 90% attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%



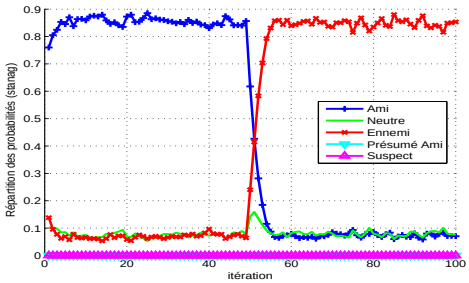
5.21.1: Résultats de la règle de Dempster



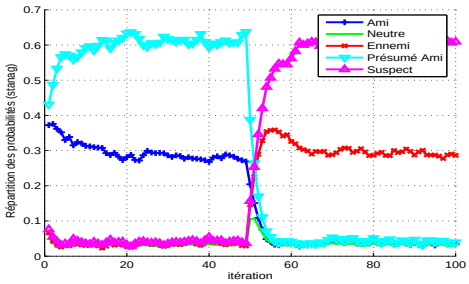
5.21.2: Résultats de la règle de DSmH



5.21.3: Résultats de la règle PCR5

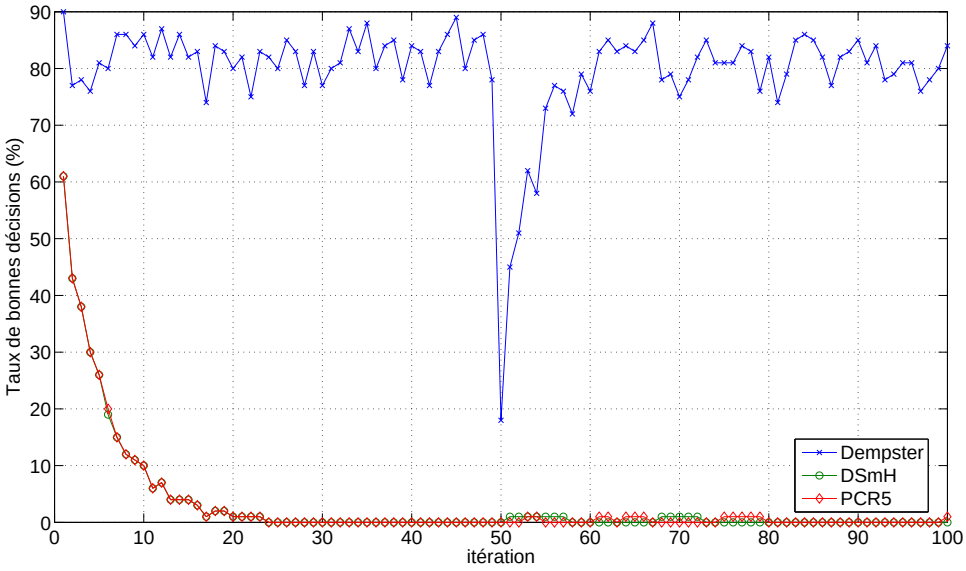


5.21.4: Résultats de la règle SACR

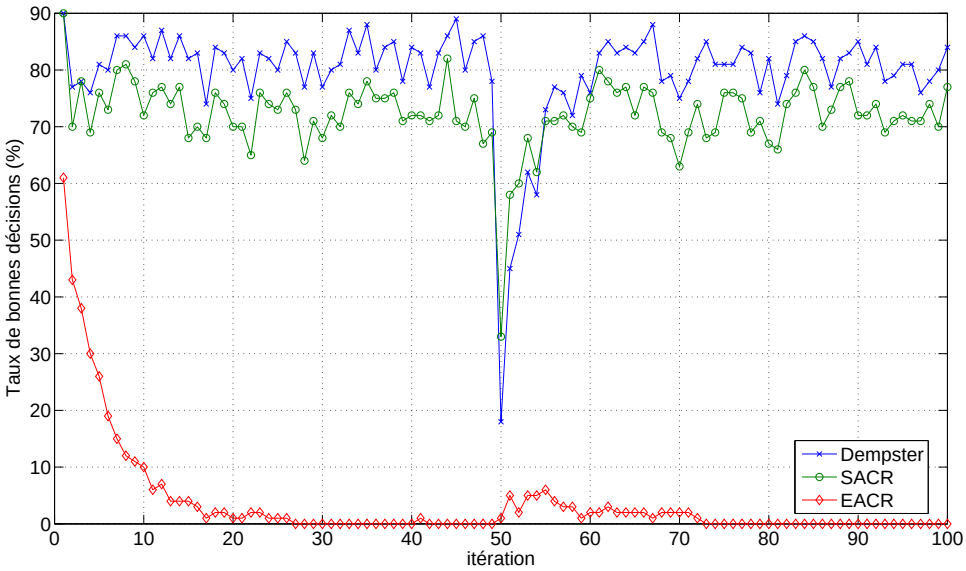


5.21.5: Résultats de la règle EACR

FIG. 5.21 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 5%

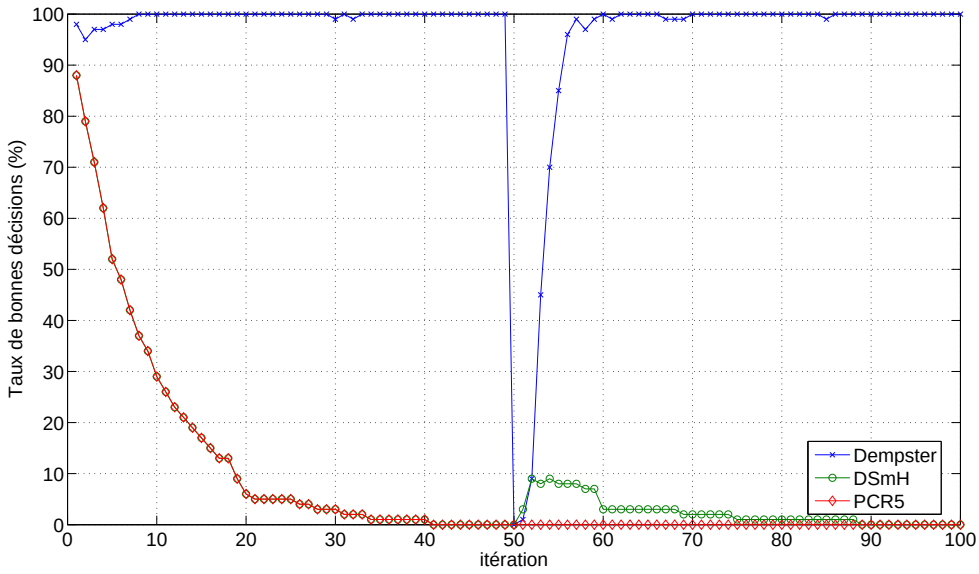


5.22.1: Règles de Dempster, DSmH, et PCR5

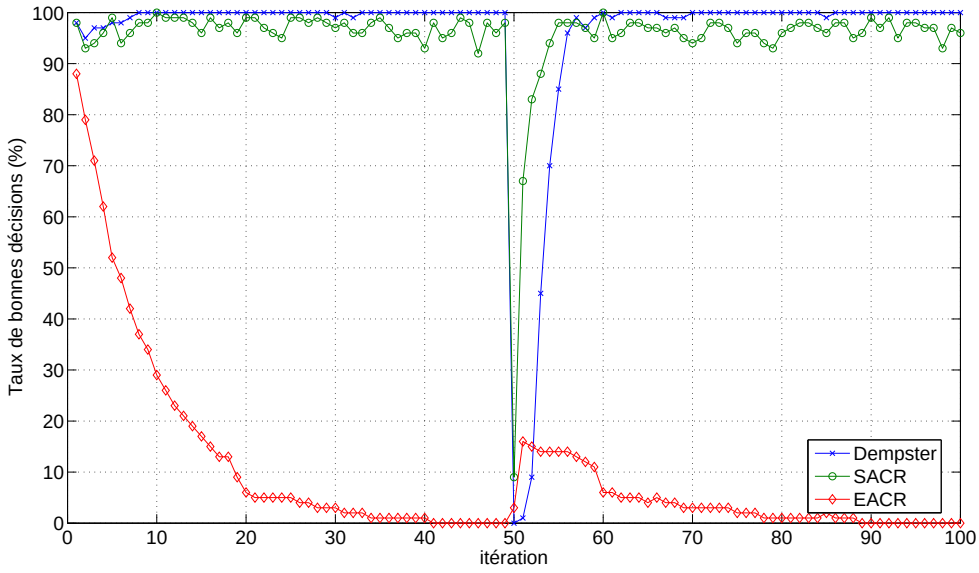


5.22.2: Règles de Dempster, SACR, et EACR

FIG. 5.22 – Taux de bonnes décisions basé sur les probabilités (stanag) avec une masse de 90% attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%



5.23.1: Règles de Dempster, DSmH, et PCR5



5.23.2: Règles de Dempster, SACR, et EACR

FIG. 5.23 – Taux de bonnes décisions basé sur les probabilités (stanag) avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 5%

5.4 Différents effets pour différentes variantes

On présente ici les effets de l'utilisation d'un algorithme de quasi-associativité, de l'arrivée de contre-mesures en salve, et enfin d'un seuil décisionnel plus ou moins élevé.

5.4.1 Effets d'un algorithme de quasi-associativité

Les simulations de la section 5.2 montrent que la quasi-associativé seule ne règle pas le problème des oscillations des fonctions de masse résultant de combinaisons.

Notez que les versions de quasi-associativité présentées et utilisées ne font qu'appliquer une règle de combinaison de base associative et applique une transformation lorsque l'on veut prendre une mesure. Cette méthode ne rend pas une règle associative, malgré qu'elle en utilise une qui l'est. La répartition de l'information suivant la procédure prescrite par la règle au moment de la prise de mesure rend la règle quasi-associative.

5.4.2 Effets de l'arrivée de contre-mesures en salve

Il faut faire bien attention. Quelle que soit la règle non associative utilisée, et ainsi, peu importe son temps de réaction, l'arrivée de contre-mesures en salve finit par leurrer tout bon raisonnement. Il faut donc, malgré tout, se méfier du résultat de combinaison. En fait, le seul moyen vraiment fiable contre les contre-mesures en salve dans un contexte où le temps est un facteur critique, serait une diversité des capteurs, des sources d'informations. Non seulement par leur position, capacité et autres paramètres, mais également sur le type même du capteur. Par exemple, lorsque l'on souhaite traverser à pied une intersection, il faut non seulement utiliser son sens auditif, mais également son sens visuel.

Le temps est un facteur clé dans cette situation. En effet, des contre-mesures apparaissant durant une courte période de temps est analogue à un début de changement d'allégeance. Si la période s'étire, une résistance aux contre-mesures devient équivalent à une mauvaise réactance aux changements d'allégeance.

5.4.3 Effets d'un seuil décisionnel plus ou moins élevé

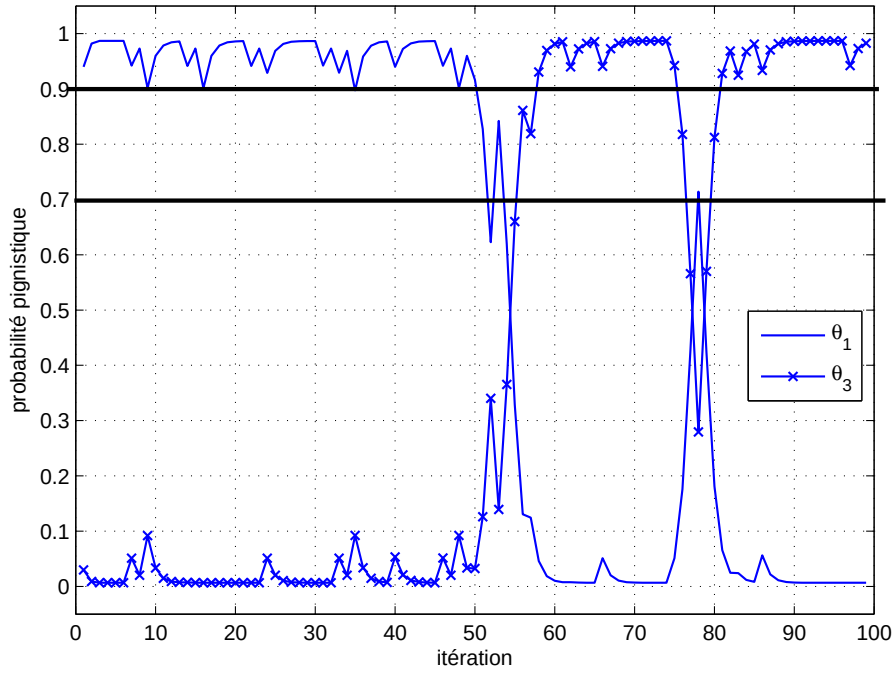
Nous reprenons ici deux cas de la section 5.2 afin de démontrer l'importance d'une bonne sélection du niveau du seuil pour décider d'une action à entreprendre. La figure 5.24.1 montre le résultat de la combinaison de la règle de Dempster et la seconde figure, la figure 5.24.2 montre le résultat de la combinaison de la règle SACR. Dans les deux cas, les sorties des deux règles ont subi une transformation pignistique généralisée.

Dans un premier temps, avec un seuil décisionnel à 0.70, la règle de Dempster prend aisément une décision juste sur toute la durée de la simulation sauf au point de changement d'allégeance et durant l'arrivée d'une série de contre-mesures aux itérations 76-79. Toutefois, en ces deux occasions, la décision passe à un autre singleton le temps d'une itération, ailleurs la décision est suspendue en attente d'informations supplémentaires.

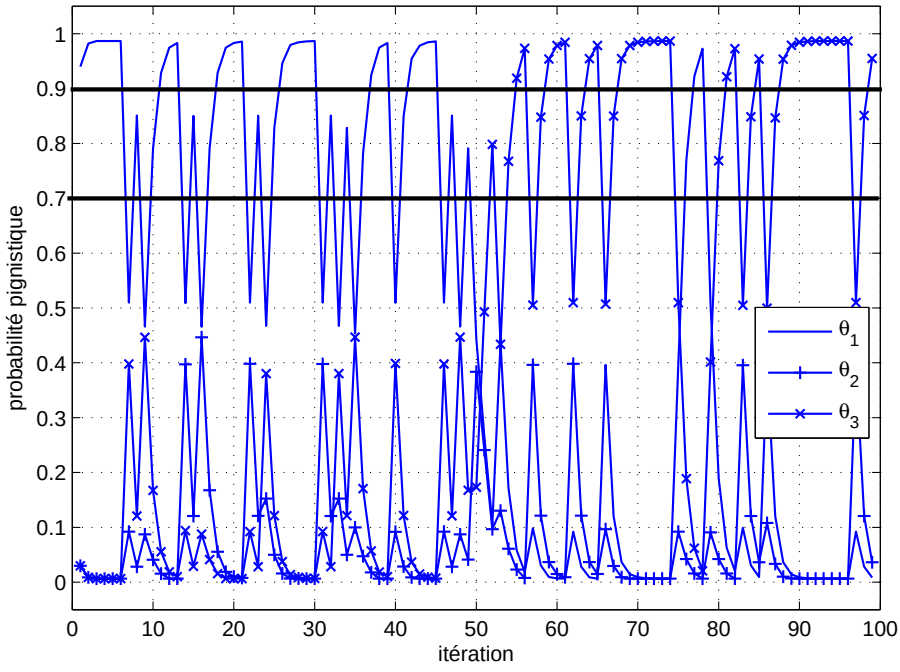
Avec ce même niveau de seuil à 0.70, la règle SACR oscille entre un singleton et l'absence brève de prise de décision. Ces absences brèves de décision ont lieu approximativement deux fois par dix itérations, ce qui est trop élevé. On note toutefois que la décision ne passe pas à un autre singleton et demeure loin des autres ensembles. Il serait donc possible de suggérer une baisse du niveau du seuil décisionnel dans ce cas précis. Prendre une décision sur une allégeance particulière alors que la combinaison donne un niveau d'incertitude élevé, constitue un risque relativement important.

Nous avons également exploré l'effet d'un seuil décisionnel à 0.90. Dans le cas de la règle de Dempster, aucun problème ne survient. On a les mêmes deux temps d'absences de décision. Cette fois-ci, ces deux temps d'absence de décision ont une durée plus élevée qu'avec le seuil de 0.70. Dans le cas de la règle SACR, une douzaine de périodes d'absence de décision sur les cent itérations de la simulation ont toutes une durée beaucoup plus élevée qu'en présence du seuil à 0.70. En fait, avec ce seuil la règle passe autant de temps sans prendre de décisions qu'à être apte à en prendre une. On en conclue que cette règle ne permet pas la prise de décision fiable avec un seuil décisionnel au-delà de 0.70.

Ainsi, le choix d'un niveau de seuil décisionnel est important et dépend de la règle de combinaison utilisée. Il faut également faire le choix en fonction de la situation et du contexte de la prise de décision. Est-on prêt à ne pas prendre de décision ? Accepterait-on de risquer de prendre une mauvaise décision ? Plusieurs questions sont à éclaircir avant la sélection du niveau du seuil décisionnel, ce choix ayant un impact direct sur la capacité de la règle à rendre de bonnes décisions.



5.24.1: Règle de DS avec filtre sur les sorties



5.24.2: Règle SACR avec filtre sur les sorties

FIG. 5.24 – Effet du niveau du seuil décisionnel sur le succès de la décision

5.5 Conclusions

5.5.1 Besoin et nécessité d'un filtre

Les figures de la section 5.3 montrent que le filtre sur les entrées améliore significativement la qualité de la combinaison par l'augmentation du taux de bonnes décisions, par la diminution de conflit lors de combinaison. Ce filtre améliore également la stabilité des décisions. (Voir théorie dans la section 4.6.2.)

5.5.2 Classement des règles selon leurs performances

Selon les critères de performance établis à la section 3.2, le classement des règles de combinaison suivant est obtenu pour les cas spécifiques explorés.

Classement sous la théorie de Dempster-Shafer

1. Règle de combinaison de Dempster
2. Règle de combinaison SACR

Classement sous la théorie de Dezert-Smarandache

1. Règle de combinaison PCR
2. Règle de combinaison DSmH
3. Règle de combinaison EACR

Si l'on souhaite une comparaison de règles dont la priorité est le fonctionnement sous le STANAG 1241, comme c'est le cas dans ce mémoire, la règle PCR est de performance égale ou supérieure à la DSmH, qui se trouve à être de performance égale ou supérieure à la EACR.

Si l'utilisation du STANAG 1241 n'est pas la priorité, mais que la priorité se trouve au niveau du taux de bonnes décisions, malgré le temps de réaction au changement d'allégeance un peu plus long, la règle de Dempster prend la première place. Selon les conditions, les règles donnent toutefois des performances très similaires.

Dans le contexte donné, les paramètres[3] suivants sont à privilégier :

[3]Notez que l'on n'a pas vraiment le contrôle sur le dernier point. Ce contrôle s'opère au niveau matériel et n'est pas toujours possible.

– présence de filtre sur les entrées,
– décision basée sur les probabilités pignistiques, et non sur les masses,
– décision basée sur les probabilités (stanag),
– ne pas utiliser la quasi-associativité, sauf celle basée sur la règle Dempster,
– seuillage de l'ignorance totale à 2%,
– taux de détection de l'*allégeance esm* par le senseur ESM d'au moins de 80%.

Notre évaluation ne tient pas compte du temps de traitement de l'information.

Chapitre 6

Conclusions

Repose-toi d'avoir bien fait, et laisse
les autres dire de toi ce qu'ils veulent.
– Phytagore

6.1 Rappel des objectifs

Notre recherche consistait à réaliser la combinaison d'informations sur l'*allégeance esm* provenant de capteurs ESM dans le cadre de raisonnement Dezert-Smarandache (DSm) (D^Θ). Pour ce faire, nous devions comparer les résultats en utilisant des règles de combinaison récentes dans DSm avec deux règles classiques, la Dempster dans la théorie de l'évidence (2^Θ) et la méthode du vote majoritaire.

6.1.1 Objectifs principaux

Nos objectifs principaux étaient donc les suivants :

1. Utiliser la DSmT pour les ESM en utilisant les définitions d'allégeance du STANAG 1241.
2. Implanter la DSmT pour un cadre de raisonnement ayant trois objets.

et cela devant être fait tout en respectant le STANAG 1241, ce qui se fait intuitivement pour la DSmT.

6.1.2 Objectifs secondaires

Nos objectifs secondaires devenaient ainsi les suivants :

1. Développer le code nécessaire pour l'utilisation de la DSmT ;
2. Valider la théorie et le code pour les ESM sous la DSmT ;
3. Adapter la théorie et le code nécessaire sous la DSmT ;
4. Comparer la règle de combinaison Dempster sous la DST et DSmH sous la DSmT pour des ESM ;
5. Tenter des optimisations au niveau de la performance opérationnelle.

6.2 Contributions du mémoire

Diverses contributions scientifiques ont été réalisées dans le cadre notre recherche. Le chapitre 4 fait la présentation détaillée de ces contributions. Elles couvrent des aspects logiciels et théoriques, pallient aux problèmes liés à la quasi-associativité, d'incohérence

de certains résultats ainsi qu'à la possibilité de fonctionnement sous DSmT. Voici donc à nouveau, en version abrégée, un tour des contributions.

Implantation logicielle

La règle de combinaison DSmH était considérée trop complexe pour une implantation. Son implantation logicielle dans le cadre de ce mémoire constitue l'une de nos réalisations. En plus, il a fallu implanter diverses règles de fusion utilisées dans le présent mémoire fonctionnant sous la DSmT.

Quasi-associativité utilisant la règle de Dempster

Nous avons suggéré et réalisé une quasi-associativité sous la DSmT qui utilise la règle de Dempster (DS) ainsi qu'une version disjonctive de la règle. Cette version disjonctive, sans justification si utilisée seule, a été pensée dans le seul but de travailler de pair avec la règle DS usuelle pour un raisonnement sous DSmT. Comme on l'a vu au chapitre 5, on obtient des résultats prometteurs avec cette quasi-associativité.

Extension à la règle de combinaison adaptative à DSmT

Un concept de conflit disjonctif a également été présenté quoique difficile à justifier. En combinant conflit disjonctif et conflit conjonctif, cela a permis de définir une combinaison adaptative étendue. Cette nouvelle règle repose à la fois sur la règle ACR et sur la règle DSmH.

Filtrage de l'information à combiner

L'utilisation d'un filtre sur les informations en entrée est également une idée nouvelle qui s'est avérée efficace. L'application d'un filtre sur les informations de sorties à également été bénéfique.

Seuillage à l'ignorance totale

L'application d'un seuil sur le niveau minimal de la masse à assigner à l'ignorance totale à chaque combinaison d'informations, constitue une solution simple au problème d'accumulation de masse vers $\cap_T$. On donne ainsi la possibilité de générer de l'information autre que l'intersection totale.

6.3 Bilan de la recherche

On considère avoir atteint les objectifs principaux (voir section 6.1.1) sans quoi aucun résultat n'aurait été obtenu dans le cadre de raisonnement D^Θ. Quant aux objectifs secondaires présentés à la section 6.1.2, le premier est atteint sans problème, il fait même l'objet d'une publication (voir [6]). Les objectifs secondaires 2 et 3 ont été réalisés théoriquement en reproduisant les exemples aux chapitres 2, 3, et 4. L'objectif secondaire 4 se retrouve réalisé grâce aux résultats présentés au chapitre 5.

Quant à l'objectif secondaire 5, le chapitre 3 et la publication [6] prouvent son accomplissement puisque la règle de combinaison DSmH a été rendue opérationnelle alors qu'elle ne l'était pas. Outre cette réalisation, des améliorations supplémentaires n'ont pas été tentées. Toutefois, quelques mesures de performances opérationnelles ont été réalisées et présentées dans [6].

Quant aux contributions apportées dans le cadre de ce mémoire, sauf celles en lien à l'implantation logicielle (voir section 6.2), elles permettent de vérifier le cadre de raisonnement DSmT dans une application pour laquelle il a des chances d'être plus approprié. On dénote plus particulièrement l'idée de la EACR et du filtre appliqué sur les données en entrée. D'ailleurs la règle EACR fera l'objet d'une publication prochainement [8]. Quelques idées ont été considérées brièvement au chapitre 4 mais qui n'ont pas été explorées, sont demeurées au stade théorique, car ils s'écartent trop de nos objectifs. Certaines idées mériteraient toutefois une attention particulière, notamment l'idée d'une redistribution du conflit considérant la fiabilité (voir section 4.5.3, page 82).

6.4 Travaux futurs

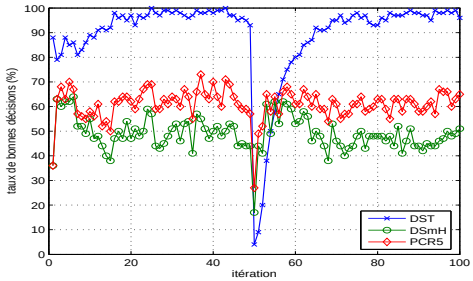
Des travaux futurs pourront avoir lieu sur la nouvelle règle de combinaison adaptative étendue (EACR). Aussi, une étude sur l'aspect des performances au niveau du temps d'exécution, des optimisations possibles, par le biais d'approximations par exemple, pourrait également avoir lieu. Il serait également intéressant d'explorer les possibilités offertes par l'idée du Cardinal DSm Généralisé Hiérarchique (CDGH). Ce dernier devrait permettre le respect de la théorie des probabilités de Bayes lorsqu'utilisé pour une transformation pignistique. La possibilité d'une combinaison dans un contexte où l'on utiliserait des objets 'virtuels' au lieu des objets qui représentent les intersections avait été brièvement considéré. Une exploration en profondeur de l'idée serait intéressante.

Il serait également intéressant d'analyser le comportement des règles de combinaison sous la DSmT mais pour d'autres contextes que celui exploré ici. Ces autres contextes pourraient être en reconnaissances des formes, ou en traitement du language, par exemple.

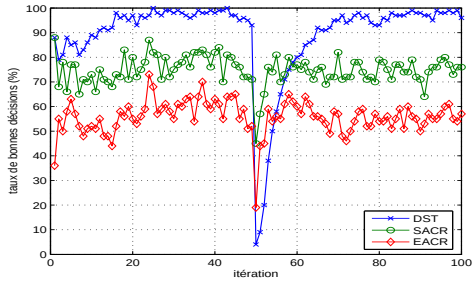
Annexe A

Résultats graphiques des simulations Monte-Carlo

Cette annexe présente les résultats de simulations Monte-Carlo. On retrouve plus exactement 8 différents cas où l'on varie les paramètres dont la masse attribuée à l'allégeance détectée, le taux de détection et le niveau de seuillage de l'ignorance totale. On retrouve les résultats graphiques sous forme de taux de bonnes décisions à partir des probabilités pignistiques pour les règles de DS, DSmH, PCR5, SACR, et EACR. On présente également la répartition des probabilités (stanag) avec les huit différents cas pour chacune des règles examinées.

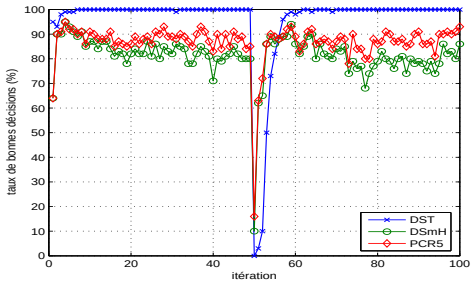


A.1.1: (P.Pign.)Dempster, DSmH, et PCR5

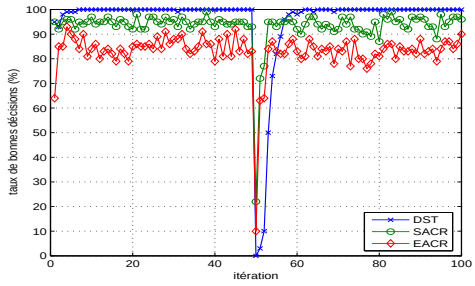


A.1.2: (P.Pign.)Dempster, SACR, et EACR

FIG. A.1 – Taux de bonnes décisions avec une masse de 70% attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%

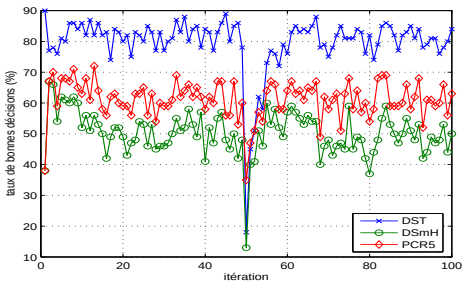


A.2.1: (P.Pign.)Dempster, DSmH, et PCR5

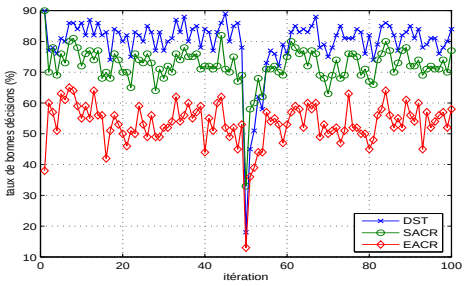


A.2.2: (P.Pign.)Dempster, SACR, et EACR

FIG. A.2 – Taux de bonnes décisions avec une masse de 70% attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 2%

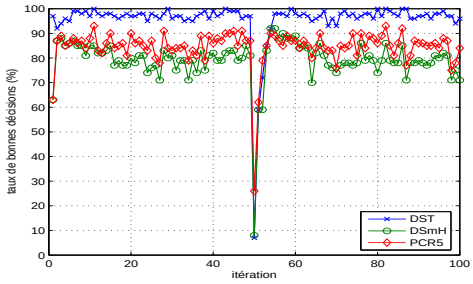


A.3.1: (P.Pign.)Dempster, DSmH, et PCR5

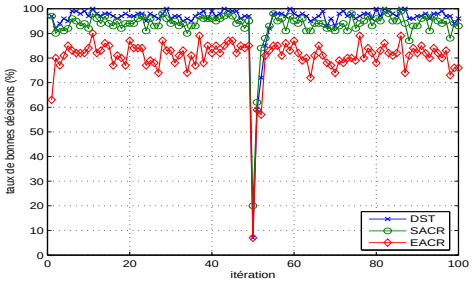


A.3.2: (P.Pign.)Dempster, SACR, et EACR

FIG. A.3 – Taux de bonnes décisions avec une masse de 90% attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%

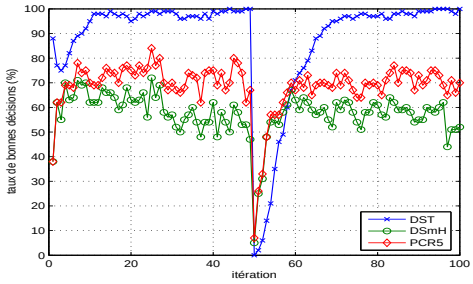


A.4.1: (P.Pign.)Dempster, DSmH, et PCR5

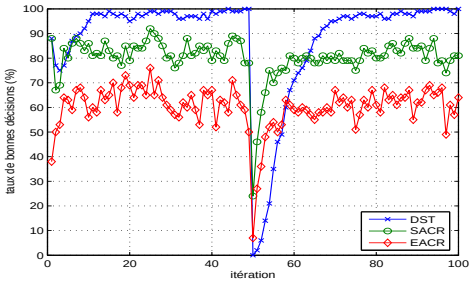


A.4.2: (P.Pign.)Dempster, SACR, et EACR

FIG. A.4 – Taux de bonnes décisions avec une masse de 90% attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 2%

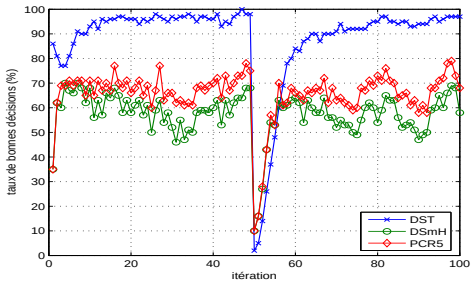


A.5.1: (P.Pign.)Dempster, DSmH, et PCR5

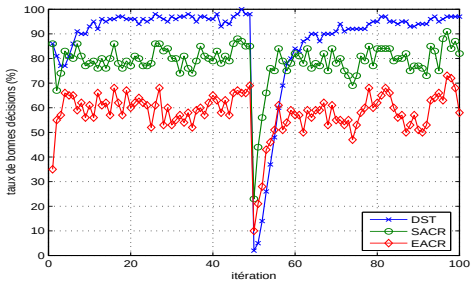


A.5.2: (P.Pign.)Dempster, SACR, et EACR

FIG. A.5 – Taux de bonnes décisions avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%

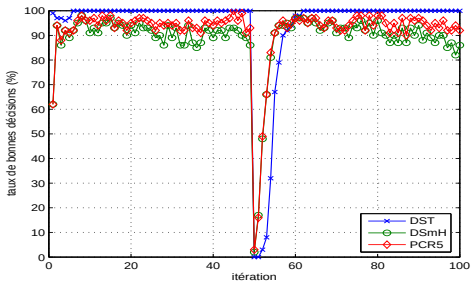


A.6.1: (P.Pign.)Dempster, DSmH, et PCR5

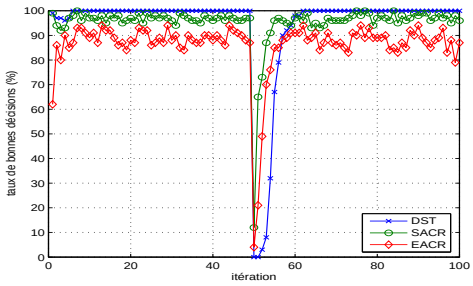


A.6.2: (P.Pign.)Dempster, SACR, et EACR

FIG. A.6 – Taux de bonnes décisions avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 5%

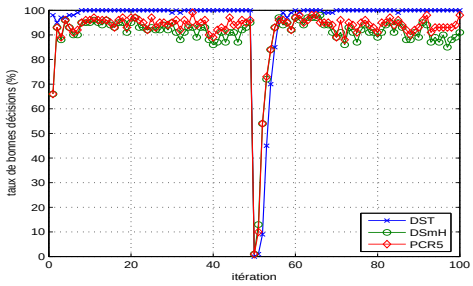


A.7.1: (P.Pign.)Dempster, DSmH, et PCR5

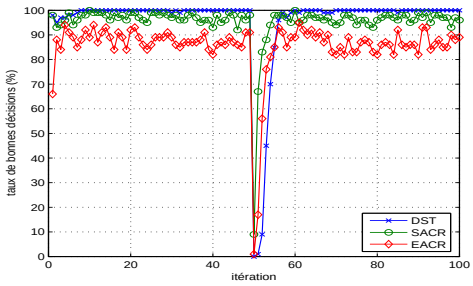


A.7.2: (P.Pign.)Dempster, SACR, et EACR

FIG. A.7 – Taux de bonnes décisions avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 2%

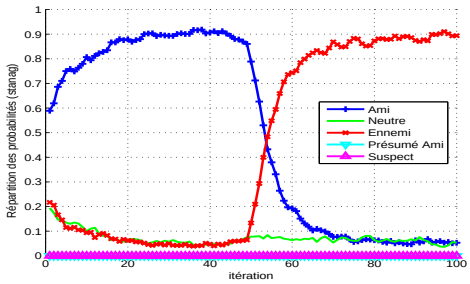


A.8.1: (P.Pign.)Dempster, DSmH, et PCR5

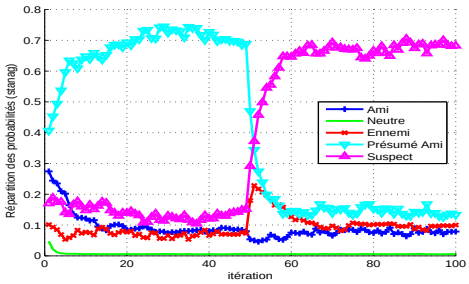


A.8.2: (P.Pign.)Dempster, SACR, et EACR

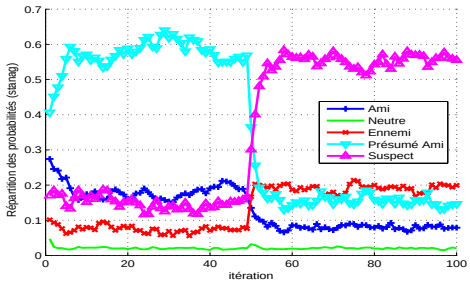
FIG. A.8 – Taux de bonnes décisions avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 5%



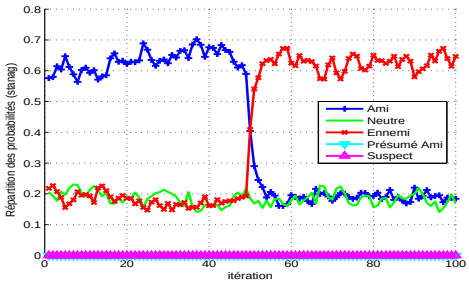
A.9.1: Règle de Dempster



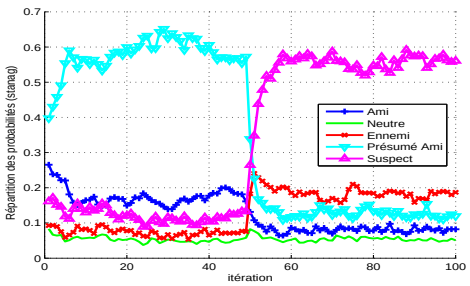
A.9.2: Règle de DSmH



A.9.3: Règle PCR5

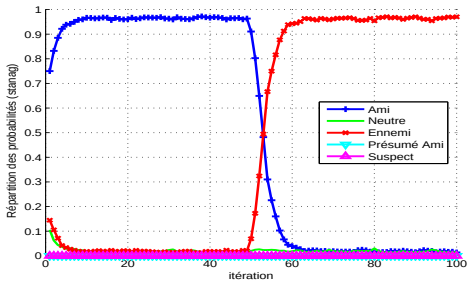


A.9.4: Règle SACR

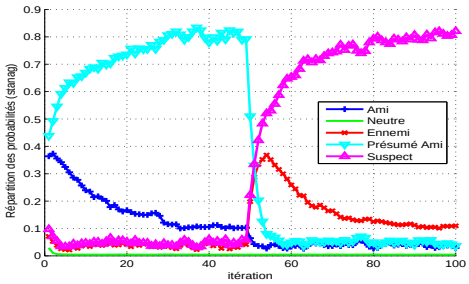


A.9.5: Règle EACR

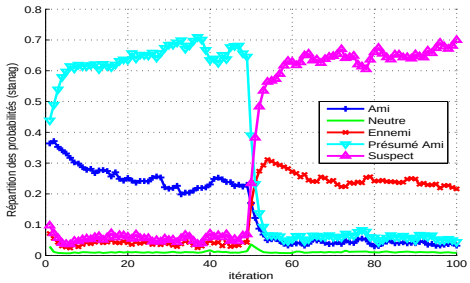
FIG. A.9 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec une masse de 70% attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%



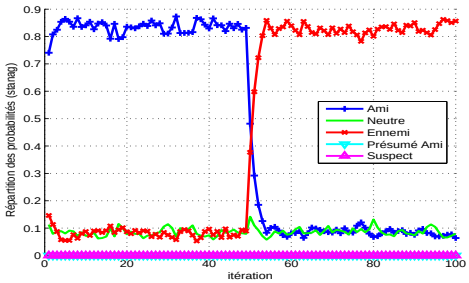
A.10.1: Règle de Dempster



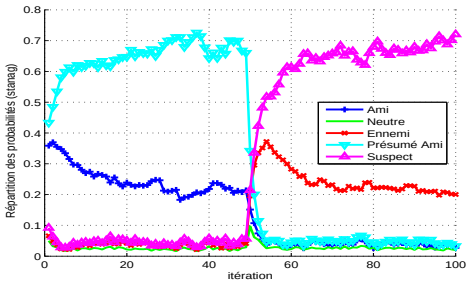
A.10.2: Règle de DSmH



A.10.3: Règle PCR5

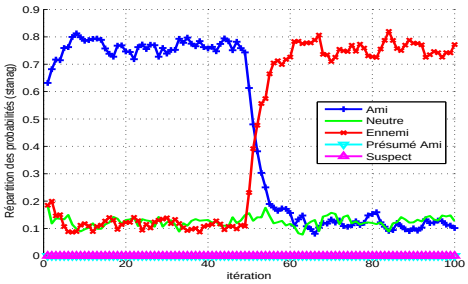


A.10.4: Règle SACR

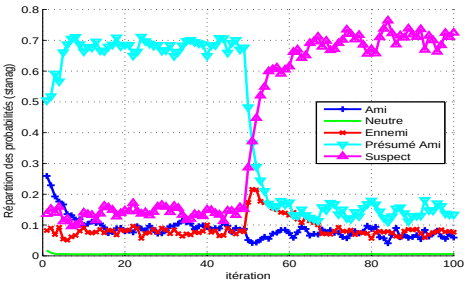


A.10.5: Règle EACR

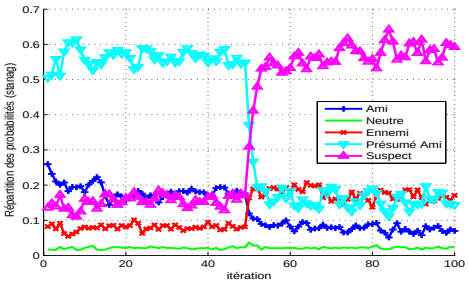
FIG. A.10 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec une masse de 70% attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 2%



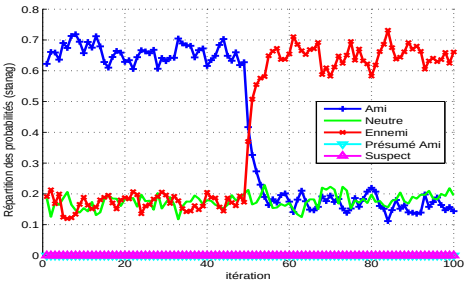
A.11.1: Règle de Dempster



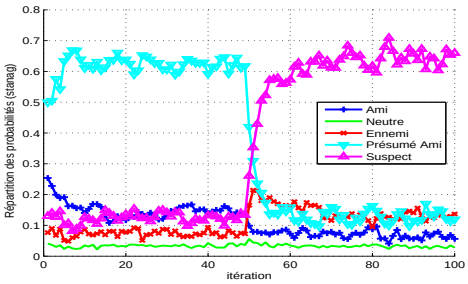
A.11.2: Règle de DSmH



A.11.3: Règle PCR5

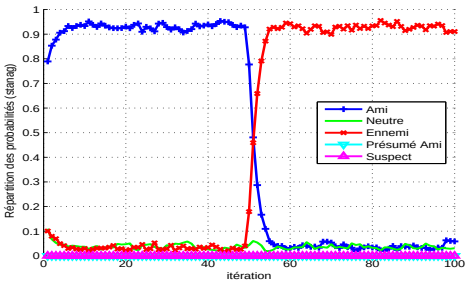


A.11.4: Règle SACR

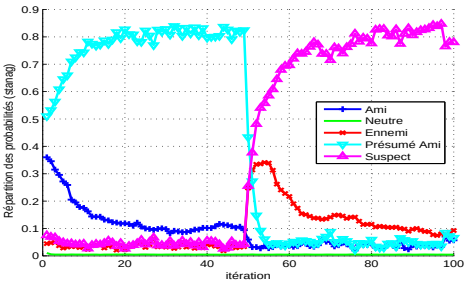


A.11.5: Règle EACR

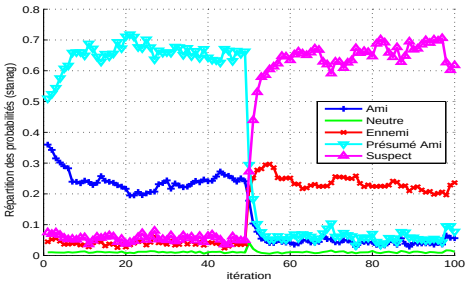
FIG. A.11 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec une masse de 90% attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%



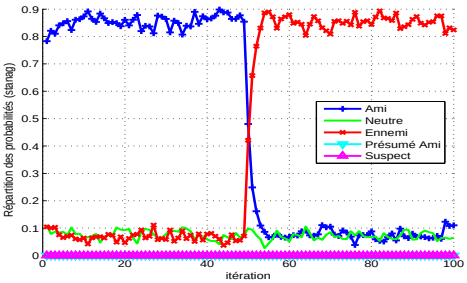
A.12.1: Règle de Dempster



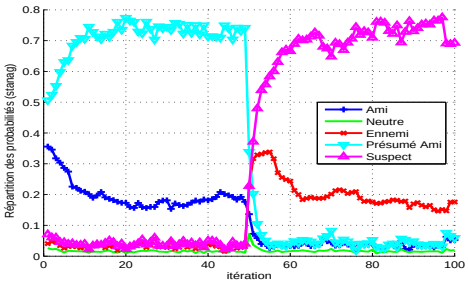
A.12.2: Règle de DSmH



A.12.3: Règle PCR5

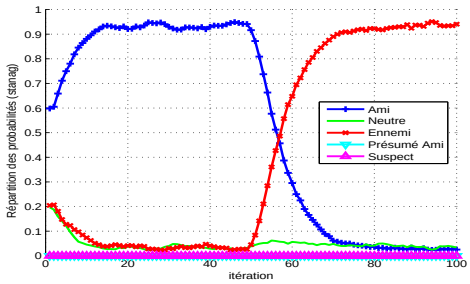


A.12.4: Règle SACR

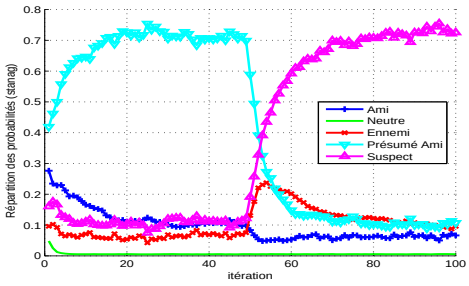


A.12.5: Règle EACR

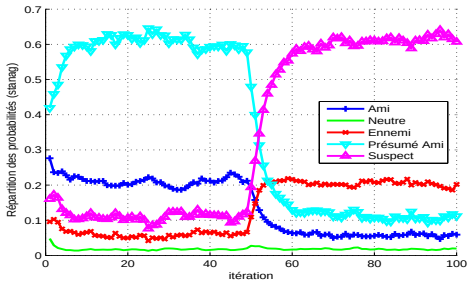
FIG. A.12 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec une masse de 90% attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 2%



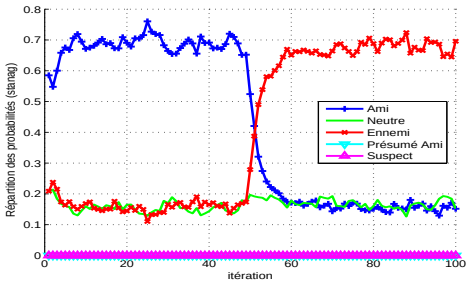
A.13.1: Règle de Dempster



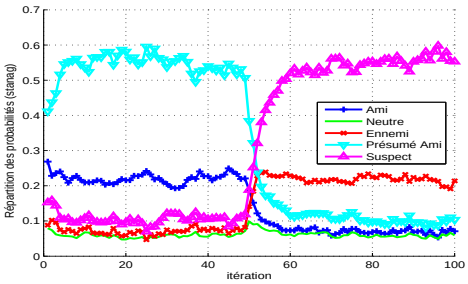
A.13.2: Règle de DSmH



A.13.3: Règle PCR5

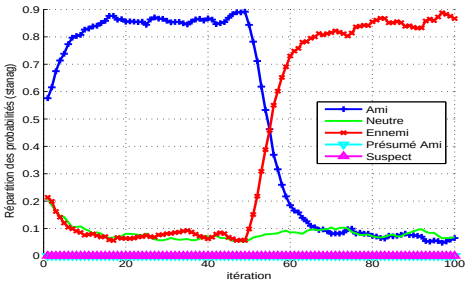


A.13.4: Règle SACR

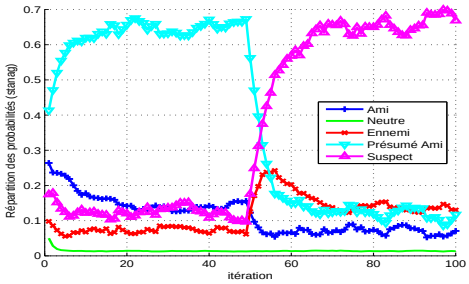


A.13.5: Règle EACR

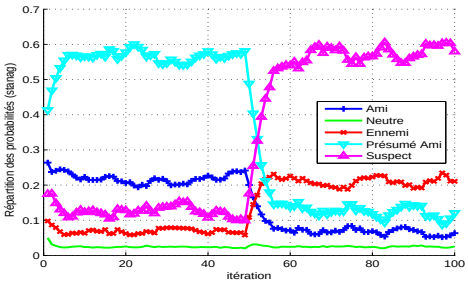
FIG. A.13 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 2%



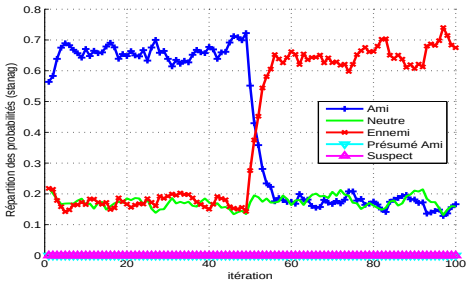
A.14.1: Règle de Dempster



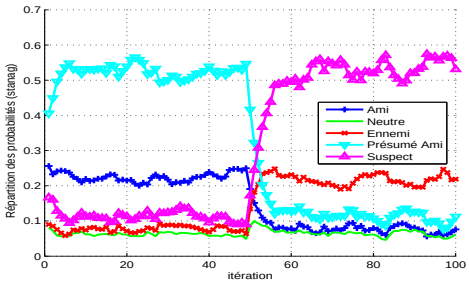
A.14.2: Règle de DSmH



A.14.3: Règle PCR5

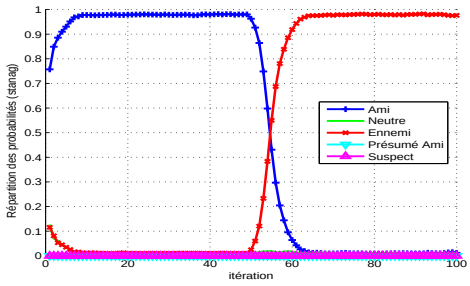


A.14.4: Règle SACR

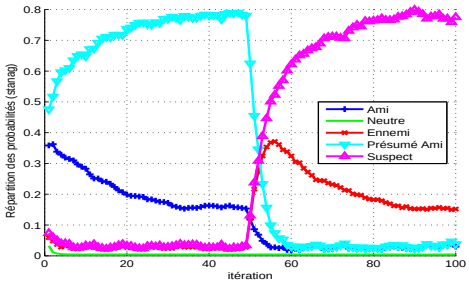


A.14.5: Règle EACR

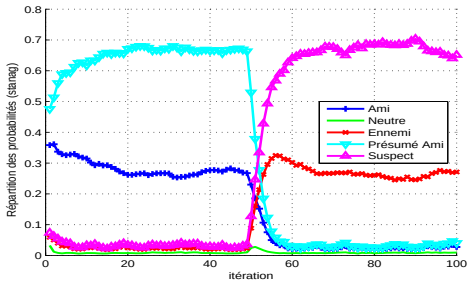
FIG. A.14 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 60% et un seuillage de I_t à 5%



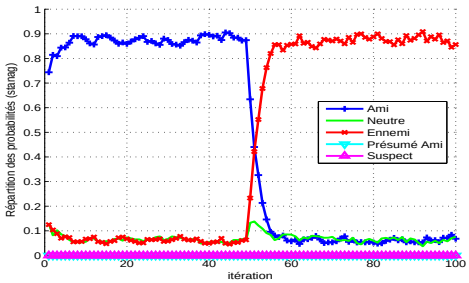
A.15.1: Règle de Dempster



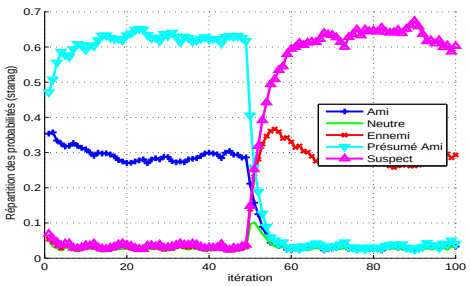
A.15.2: Règle de DSmH



A.15.3: Règle PCR5

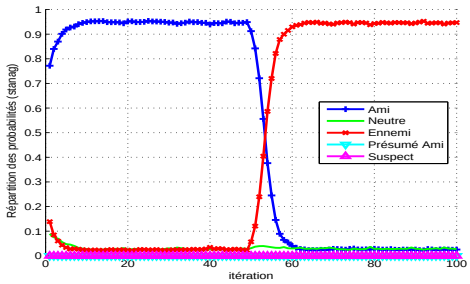


A.15.4: Règle SACR

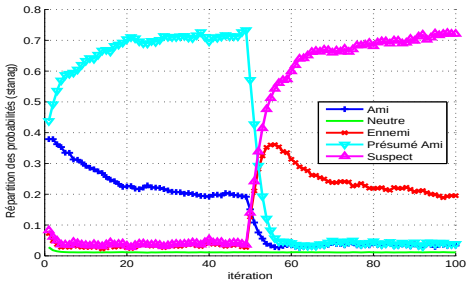


A.15.5: Règle EACR

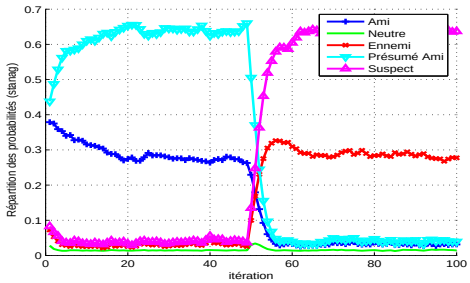
FIG. A.15 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 2%



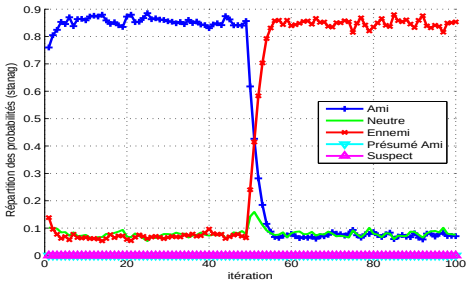
A.16.1: Règle de Dempster



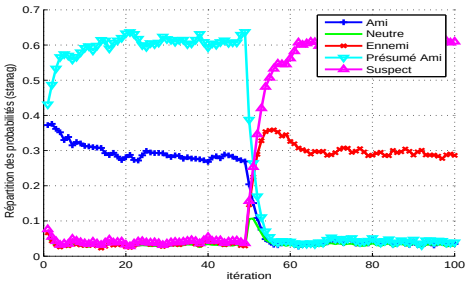
A.16.2: Règle de DSmH



A.16.3: Règle PCR5



A.16.4: Règle SACR



A.16.5: Règle EACR

FIG. A.16 – Répartition des probabilités (stanag) pour différentes règles de combinaisons avec un filtre sur les valeurs de masse attribuée à l'*allégeance esm* détectée, un taux de détection de 80% et un seuillage de I_t à 5%

Annexe B

Publication 1

Article publié à la conférence Fusion 2007 de l'ISIF.

Analysis of information fusion combining rules under the DSm theory using ESM inputs

Pascal Djiknavorian Dominic Grenier
Département de Génie Électrique et Informatique
Faculté des Sciences et de Génie, Université Laval
Québec, Canada, G1K 7P4
Email: dominic.grenier@gel.ulaval.ca

Pierre Valin
Decision Support Systems section
Defense R&D Canada Valcartier
Québec, Canada, G3J 1X5
Email: pierre.valin@drdc-rddc.gc.ca

***Abstract*— In the context of Electronic Support Measures, the use of the Dempster-Shafer Theory is not flexible enough to obtain a clear evaluation of the state of allegiance for a detected target. With the new theory of plausible, paradoxical, and neutrosophic reasoning, the Dezert-Smarandache Theory, we are able to get a clearer assessment. The current paper presents our research for these cases.**

Keywords: Dempster-Shafer, Dezert-Smarandache, Combining rules, Electronic support measures.

I. Introduction

As introduced in [1], the Dezert-Smarandache Theory (DSmT) is able to combine information even in the presence of large conflicts and constraints. Not only does the DSmT circumvent the well known problems of the Dempster-Shafer Theory (DST) [2] reported by Zadeh in [3], but it can also help us get a more precise assessment of the possible truth as we will see in section (I-B.1). Even if the DSmT resolves Zadeh's problem with the DST, one still faces a problem with the DSmT which was exposed in [4], namely that combinations under DSmT were too complex. However, this was not true anymore after the work presented in [5]. We are now able to easily experiment with the DSmT in different cases never explored before due to its apparent complexity.

As mentioned in the abstract, the current paper will present our work on an analysis between the DST and the DSmT in a ESM input environment. We will begin by a brief review of concepts required for the paper in this section, followed by theorical adaptations and the main aspects of the implementation process in section (II). We then present a selection of results and decisions taken by the implementation for a representative example in section (III). The final section presents our conclusions and a review of our results and possible future research with the DSmT.

A. Combining rules for data fusion

1) Definitions:

- *Basic Set* (Θ) : $\Theta = \{\theta_1, \theta_2, \dots \theta_n\}$. It's the set including every possible object θ_i. This set is exhaustive and its elements are not exclusive.
- *Power set* $\left(2^\Theta\right)$: represents the set of all possible sets using the objects (singletons) of the basic set Θ. It includes the empty set and excludes intersections. With the basic set defined above, we get the power set:

$$2^\Theta = \{\emptyset, \{\theta_1\}, \{\theta_2\}, \dots \{\theta_n\}, \{\theta_1, \theta_2\},$$

$$\dots \{\theta_1, \theta_2, \dots, \theta_n\}, \dots, \Theta\}.$$

- *Hyper-power set* $\left(D^\Theta\right)$: represents the set of all possible sets using the objects of the basic set Θ and allowing intersections between singletons. It includes the empty set. With the basic set: $\Theta = \{\theta_1, \theta_2\}$, we get the hyper-power set

$$D^\Theta = \{\emptyset, \{\theta_1\}, \{\theta_2\}, \{\theta_1 \cap \theta_2\}, \{\theta_1 \cup \theta_2\}\}.$$

- *Conjunctive Power set* $\left(2^\Theta_\cap\right)$: represents the set of all possible sets using the objects of Θ. It includes the empty set and excludes disjunctions. Defined as a mathematical object, it helps for the evaluation of the DSm cardinal. With the basic set $\{\theta_1, \theta_2, \dots \theta_n\}$, we get the conjunctive power set:

$$2^\Theta_\cap = \{\emptyset, \{\theta_1\}, \{\theta_2\}, \dots \{\theta_n\}, \{\theta_1 \cap \theta_2\},$$

$$\dots \{\theta_1 \cap \theta_2, \dots \cap \theta_n\}\}.$$

- *Constraint* : is a set considered impossible to obtain.
- *Constraints power set* $\left(D^\Theta_c\right)$: is the set containing all elements considered as constrained in the D^Θ.
- *Basic belief assignment* (bba) : $m : 2^\Theta \rightarrow [0, 1]$, so the mass given to a set $A \subseteq \Theta$ obeying $m(A) \in [0, 1]$.
- *Core of* Θ $(\mathcal{K})$: The set of all focal elements of Θ, where a focal element is a subset A of Θ such that $m(A) > 0$.

2) Conjunctive combining rule [7]: When we refer to the conjunctive combining rule, we will refer to the version described as:

$$q(A) = m_1 \wedge m_2 = \sum_{\substack{B \cap C = A \\ B, C \subseteq \Theta}} m_1(B)\, m_2(C) \quad \forall A \subseteq \Theta. \quad (1)$$

3) Disjunctive combining rule [8]: When we refer to the disjunctive combining rule, we will refer to the version described as:

$$q(A) = m_1 \vee m_2 = \sum_{\substack{B \cup C = A \\ B,C \subseteq \Theta}} m_1(B)\, m_2(C) \quad \forall A \subseteq \Theta. \quad (2)$$

4) Dempster-Shafer Theory (DST): The DST rule of combination is a conjunctive normalized rule working on the power set. It combines information with intersections. This theory works with the hypothesis of mathematically independent sources of evidence. The i^{th} bba's source of evidence is denoted m_i. The DST works within 2^{Θ}.

Equation (3) describes the DST rule of combination where K is the conflict and is $\forall C \subseteq \Theta$, the conflict in DST being defined by equation (4).

$$(m_1 \oplus m_2)(C) = \frac{1}{1-K} \sum_{A \cap B = C} m_1(A)\, m_2(B) \quad (3)$$

$$K = \sum_{A \cap B = \emptyset} m_1(A)\, m_2(B) \qquad A, B \subseteq \Theta \quad (4)$$

5) Dezert-Smarandache Theory (DSmT): The DSmT uses the hyper-power set being thus able to work with intersections. The DSmT possesses two rules of combination which are able to work around the mass redistribution problem of the DST in the presence of large conflicts:

- The Classical DSm rule of combination (DSmC) is based on the free model $M^f(\Theta)$

$$m(C) = \sum_{A \cap B = C} m_1(A)\, m_2(B) \quad \underset{\forall C \in D^{\Theta}}{A, B \in D^{\Theta}} \quad (5)$$

- The Hybrid DSm rule of combination (DSmH) is able to work with many types of constraints

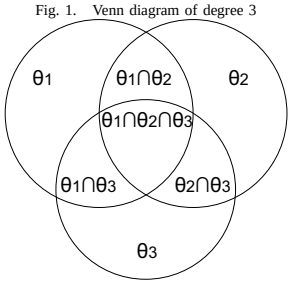
$$m_{M(\Theta)}(A) = \phi(A)\left[S_1(A) + S_2(A) + S_3(A)\right] \quad (6)$$

$$S_1(A) = \sum_{X_1 \cap X_2 = A} m_1(X_1)\, m_2(X_2) \quad \forall X_1, X_2 \in D^{\Theta} \quad (7)$$

$$S_2(A) = \sum_{\substack{[(u(X_1) \cup u(X_2)) = A] \vee \\ [((u(X_1) \cup u(X_2)) \in \emptyset) \wedge (A = I_t)] \\ \forall X_1, X_2 \in \emptyset}} m_1(X_1)\, m_2(X_2) \quad (8)$$

$$S_3(A) = \sum_{\substack{X_1 \cup X_2 = A \\ X_1 \cap X_2 \in \emptyset}} m_1(X_1)\, m_2(X_2) \quad \forall X_1, X_2 \in D^{\Theta} \quad (9)$$

Fig. 1. Venn diagram of degree 3



Note that $\phi(A)$ in equation (6) is a binary function resulting in 0 for empty or impossible sets, and 1 otherwise. In equation (8), $u(X)$ represents the union of all objects of set X. Equation (9) is the union of all objects of sets X_1 and X_2, when it is not empty. From equation (8), I_t represents the total ignorance, or the union of all objects part of the basic set.

The DSmH can also be viewed in an incremental way: **Step S1**: if $(\theta_1 \cap \theta_2 \in D_c^{\Theta})$, then continue to step S3, otherwise the mass $m_1(X_1)\, m_2(X_2)$ is added to the mass of $A = (\theta_1 \cap \theta_2)$. **Step S3**: if $\left(\theta_1 \cup \theta_2 \in D_c^{\Theta}\right)$, then continue to step S2, otherwise, the mass $m_1(X_1)\, m_2(X_2)$ is added to the mass $A = (\theta_1 \cup \theta_2)$. **Step S2**: if $\left(u(X_1) \cup u(X_2) \in D_c^{\Theta}\right)$, then add the mass to I_t, otherwise, the mass $m_1(X_1)\, m_2(X_2)$ is added to the mass of $A = (u(X_1) \cup u(X_2))$.

Complete examples of the DSmT are available in [6].

B. Electronic Support Measures (ESM)

An ESM is a passive sensor that captures incoming electromagnetic energy, which after treatment, reveals informations about a detected target, such as direction (bearing), identity, and allegiance of the target that emitted the radiation, together with a belief about the identity and allegiance.

The belief information thus received is interpreted as a mass for each of the three possible ESM declarations which are friendly, neutral and hostile. In the following we will describe them as θ_1 (friendly), θ_2 (neutral) and θ_3 (hostile). Figure (1) presents us the Venn diagram for a case without any constraints.

While the ESM only gives us 3 choices for the allegiance, the decision maker requires 5 possibilities for the allegiance (according to STANAG 1241 - NATO Standardization Agreement), which, in the natural language of the DSmT makes $\theta_1 \cap \theta_2$ correspond to *assumed friendly*, with the other intersections corresponding to *suspect* (involving an intersection with θ_3).

1) DST precision problem: Figure (2) shows us the extent of the limitation of the DST in our case showing it's inability to specify when a target is 'assumed friendly' or 'suspect'. It is equivalent to take $D_c^{\Theta} = D^{\Theta} \backslash 2^{\Theta}$, then all intersections are constraints. Without this ability systems with DST will directly switch between allegiance states avoiding some allegiance possibilities. These inaccessible states can be accessed using the DSmT.

Fig. 2. Venn diagram of degree 3 for DST

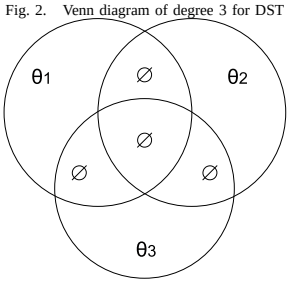
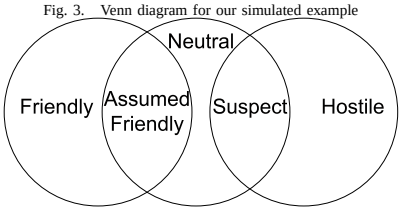


Fig. 3. Venn diagram for our simulated example



II. Theory and implementation

The main challenges and difficulties encountered in this research are found on the level of the effective, or optimal, implementation of the DSmH combining rule. In the present section, we will review critical points where an original reasoning allowed the implementation of a combining rule originally considered too complex [4], and of the implementation of the Generalized Pignistic Transformation (GPT). There is also the issue of the oscillations which is addressed by two different tools, the first one being the Florea's Quasi-Associativity method [9], the second one being a filter.

A. Constraints on simulation

Our simulation, as mentioned earlier, works with the basic set described by figure (1), however we've added constraints on sets $\{[\theta_1 \cap \theta_3], [\theta_1 \cap \theta_2 \cap \theta_3]\}$. So the simulated example's Venn diagram is represented by figure (3).

B. Combining rules

1) Generated ESM information: We seek to generate as realistic ESM declarations as possible. The situation that we have in mind is one where the ESM declaration could have been caused by multiple targets trying to hide behind each other within the angular accuracy of the ESM's bearing measurement. For example, if one were to track a friendly target, ESM reports from any target within the bearing accuracy of the ESM's bearing measurement could be associated to our friendly. Thus on average we could correctly associate a friendly ESM declaration (say) 70% of the time in a dense environment, with 15% sometimes resulting from miss-association of a "neutral", and "15% from a "hostile", presumably all close in bearing, but far away in range.

Thus we have used a randomly generated ESM declaration with a probability that the object θ_1 is selected 70% of the time for the first 50 iterations, out of 100 iterations (with the remaining 30% split evenly between the other two possibilities). From the 51^{th} iteration to the last one, the object θ_3 is selected 70% of the time (again with the remaining 30% split evenly between the other two possibilities). For each new ESM declaration, the selected allegiance is given the mass of 0.70, with the rest of the mass being given to total ignorance (I_t). In splitting up each Monte-Carlo run at the halfway point, we can see the capability of each rule in identifying a true change in allegiance.

2) Combining rule's results: To insure a certain level of computational safety, and to avoid the DST problem when the conflict approaches the value of 1, we've applied a filter at the output of all combining rules. That filter insures that the mass given to the total ignorance (I_t) is never less than 2%.

3) Data format: The information is contained in the same way described in [5]. Thus, we keep in memory only the core of the bba of the BOEs (Bodies of Evidence). Each core information is kept in a structure containing two objects, one being the matrix of objects, the other one being the vector of masses. Each one of the objects is kept as a product of sums, intersections are kept as vectors, unions are composed of multiple cells in a matrix, each cell being a vector (intersection). Obviously the structure can be recursive to support larger objects.

4) Reasoning frame generation: The generation of 2^Θ and D^Θ is not required for the evaluation of the combining rules, but are required for the evaluation of the generalized pignistic transformation, so we have developed a way of generating 2^Θ, and from it, D^Θ. However, the generation of D^Θ work only for Θ, with $|\Theta| = 3$, which is enough for our case.

To generate 2^Θ, we built a function that uses as parameter the cardinal of Θ (3, in our case) and a factor. That factor tells us which type of power set we want, 2^Θ or $2^\Theta_\cap$. We use that function to generate both, then take their intersection and union to get D^Θ. The cardinal sets the dimension of a matrix of binary numbers that we generate in numerical ascending order.

A second matrix is built, for which each line represents an object. Each odd column is a singleton, and each even column contains a code for a conjunction or a disjunction symbol. The value contained in each odd column is a numerical value coded by taking the value of a counter. That counter, counts the number of odd columns and resets when changing line. To know when to put in a value for each position, the function reads the matrix of binary numbers. In that matrix, the line n corresponds the object represented by the line n in the second matrix. The column m of the matrix of binary numbers represents the singleton θ_m.

After the second matrix is built, a treatment is made to merge columns, put the objects in the appropriate format, sort the information, and attribute temporary mass values.

5) Combining rule implementation: The implementation of rules uses the system developed and presented in [5].

C. Classical Pignistic Transformation

The Classical Pignistic Transformation (CPT) is described in [7] and modified in [1] is represented by:

$$\Pr\{A\} = \sum_{X \in 2^\Theta} \frac{|X \cap A|}{|X|} m(X), \forall A \in 2^\Theta. \quad (10)$$

In equation (10), $|X| = n$ represents the classical cardinality of a set X, and n, the number of elements in the set. The CPT can be used to transform the bba into a probability function in cases within 2^Θ.

D. Generalized Pignistic Transformation

The Generalized Pignistic Transformation (GPT) as described in [1] required modifications so its implementation would be efficient. If one takes a look at the original equation for the GPT, in section (7.3) of [1] we have:

$$\Pr\{A\} = \sum_{X \in D^\Theta} \frac{C_\mathcal{M}(X \cap A)}{C_\mathcal{M}(X)} m(X), \forall A \in D^\Theta. \quad (11)$$

For efficiency purposes, even if A is considered as a set part of D^Θ, only $A \in 2^\Theta_\cap$ are used. This choice has been made to avoid all the redundancy of evaluating pignistic probabilities for all the sets part of D^Θ, considering that we can evaluate disjunctions by simple summations and subtractions as in the probability theory. Theses evaluations can be accomplished using Theorem 2.3.4 as seen in [10] (See example case in equation (12)). It also had the effect of optimizing our simulation system.

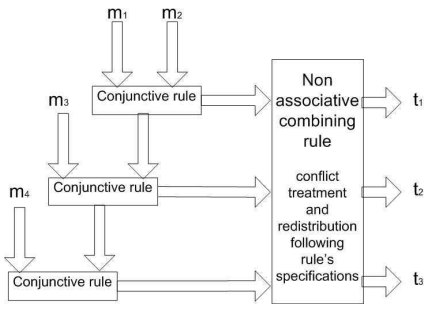
$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \quad (12)$$

For the same reason of efficiency, we've modified the range of X. Rather than $X \in D^\Theta$, which is true, we have restricted it to the core, $\mathcal{K}$. Hence, it will be for the subset of D^Θ which includes only non zero masses. In other words, only the sets given in input will be considered, since our system considers only non empty objects. Note that the GPT requires to take the decision only by choosing the object with the maximum pignistic probability.

1) DSm cardinal: The DSm cardinal could be viewed as presented in chapter 3 of [1]. It can also be viewed and implemented as in the following algorithm. For E, $C_\mathcal{M}(E)$ can be considered as the sum of the number of effective intersections with F, $\forall F \in D^\Theta_*$, where D^Θ_* is the set D^Θ without constraints ($D^\Theta_* = D^\Theta \backslash D^\Theta_c$).

2) GPT implementation: We have implemented the GPT as a function which takes into parameters the current core of Θ and current constraints and then gives us a structure containing a matrix of objects associated with a vector of computed pignistic probabilities. The function evaluates at each call DSm cardinal values for all possible sets of D^Θ_*, and then proceeds with equation (11). When required, a value of any given DSm cardinal is not evaluated but looked up in the table of DSm cardinal values previously built.

Fig. 4. Florea's Quasi-Associativity



3) Power set cases resolved under the GPT: Our system has only the GPT version of pignistic transformation implemented. This is quite sufficient, since as demonstrated in both examples of section (7.4) of [1], the GPT is able to work with Shafer's model cases considering $D^\Theta_c = D^\Theta \backslash 2^\Theta$. Since our system works under D^Θ, we need to add to the list of constraints the list of all possible conjunctions so the GPT would work as the CPT.

E. Quasi-Associativity (QA)

In our implementation, we've also experimented with an algorithm to implement QA into DSmT. We've experimented with the QA version presented in [9] and a modified version. Florea's algorithm exploits the fact that the conjunctive and disjunctive rules respects the associativity property.

As we see in figure (4), Florea's QA combines the information using the conjunctive rule until time (iteration) τ where we have to make a decision. At that time, we use the new information from time τ and combine it using the non-associative rule with the combined output of the conjunctive rule of time $(\tau - 1)$. Thus turning a non-associative rule to a QA rule.

In [9], it is only shown that the result of such QA rule respect the associativity property. As shown in figure (4), the result of the QA rule at time τ is not used for subsequent combinations, it is the result of the conjunctive rule that is used in subsequently.

Two adaptations were developed from Florea's QA for our research. Since the disjunctive rule converges too quickly toward total ignorance (I_t), it cannot be used as described in [9] with both conjunction and disjunction as it would be required by DSmT. So the first adaptation we had to make, is the use of the conjunctive rule only. The second adaptation is a modification to the first one. This time, instead of the conjunctive rule, we'll be using the DST's combining rule.

1) QA implementation: Our implementation of Florea's QA is not exactly as described by figure (4). The difference is at the point where the non-associative combining rule takes it's input information. Florea's version, takes the input information from the output of the conjunction at time τ, and combines

it to get the output at time τ. In our version, since the non-associative combining rule version has it's own conjunction calculator module, we've taken the combined output of the conjunctive rule of time $(\tau-1)$ as an input. The implemented system can thus be, and stay, modular and scalable.

F. Filters and preconditioning filter

To avoid an output with a behavior having oscillations, we've tried different types of filters. The first one is rather simple, but doesn't have a theoretical basis other than smoothing the graphical output. The filter applies an arithmetic mean on a window of the output. In a second experimented filter is a little more original, we've tried a filtering by window method on the input mass information. Theoretically, this is based on the hypothesis that the oscillations present in the output may be caused by unprecise, and/or biased input. Thus, input filtering would attenuate the effect of incorrect inputs.

1) Filters implementation: The implementation of the first type of filter, as described in the previous section, is based on the following equation:

$$m_\tau = \frac{\left(\frac{m_{\tau-3}+m_{\tau-2}+m_{\tau-1}}{3}\right)+m_{\tau 0}}{2}, \qquad (13)$$

where $m_{\tau 0}$ is the mass at time τ of the input, and m_τ is the filtered mass for time τ. The filter form was chosen arbitrarily to smoothen the graphical output. We could've tried many types of filter that kept coherent results and let the rule react in a promptly to new information. One of the possibility was to take a continuous window with an exponential weighting, instead of a discrete window.

The implementation of the second type of filter counts the number of occurrences of each of the possible singletons in the last four iterations including the current one. If the current singleton θ_a occurred more than 3 times out of the 4 possibilities, the mass of the bba of the BOE is kept intact. If θ_a occurred only 2 times, it's mass is reduced from 0.70 to 0.60, the balance always going to total ignorance (uncertainty). If θ_a occurs only 1 time, the given mass for θ_a is 0.50, thereby giving more mass to uncertainty, since the input information hasn't proven to be of high quality. Note that we shouldn't reduce the mass of θ_a to values lower than 0.5 since we don't want to give a mass too high for the ignorance.

III. Results and Decision

This section presents us graphical outputs for a randomly generated case where we obtained a rate of occurrence of 42.57% for both θ_1 and θ_3 (close to the expected $(70+15)/2\%$), with θ_2 having the rest of the occurrences (14.85%). Figure (5) shows us which singleton was detected at which time index. Note that in this section, all figures (5-13) have the X-coordinate representing time index.

A. Cases using the Dempster-Shafer combining rule

1) Decision based on the DS rule: The figure (6) shows the mass of θ_1, $(m(\theta_1))$, as being the solid line. We can see from figure (6) that Dempster-Shafer combination rule has a fairly quick reaction when new input information arrives. Less than ten iterations after the change of allegiance toward θ_3, the dashed-line $(m(\theta_3))$ becomes dominant. At the same time, $m(\theta_1)$ decreases to lower values. No mass ever reaches the value of 1 because of the filter which guarantees a minimum mass of 0.02 to total ignorance (I_t). However, even if the combining result never reaches 1, it gives us a direction toward a specific set with confidence.

2) Decision based on the GPT with DS rule: The dash-dot line, in figure (7), representing the pignistic probability of θ_2 $(\mathrm{Pr}(\theta_2))$, is relatively important through all the time index (which might possibly become a problem in other cases). The solid-line represents $\mathrm{Pr}(\theta_1)$ and the dashed one, $\mathrm{Pr}(\theta_3)$. We

Fig. 5. Occurrences of $(\theta_1, \theta_2, \theta_3)$ from the ESM

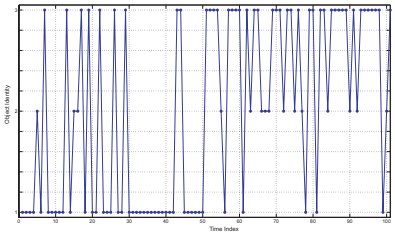


Fig. 6. $m(\theta_1)$ and $m(\theta_3)$ for DS rule

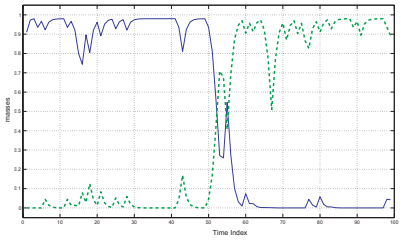


Fig. 7. Pignistic probabilities for $(\theta_1, \theta_2, \theta_3)$ for GPT DS rule

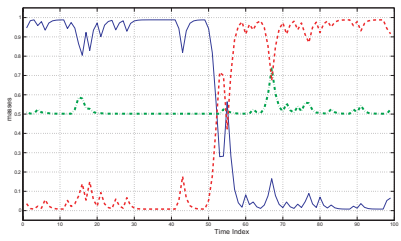


Fig. 8. $m(\theta_1)$ and $m(\theta_3)$ for DSmH

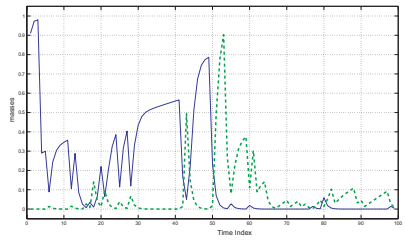


Fig. 9. $m(\theta_1 \cap \theta_2)$ and $m(\theta_2 \cap \theta_3)$ for DSmH

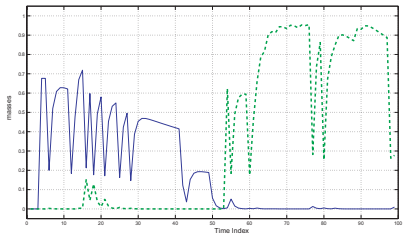


Fig. 10. Filtered sum of masses including θ_1 and θ_3 for DSmH

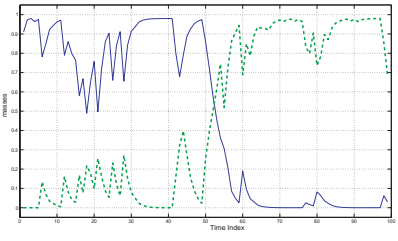
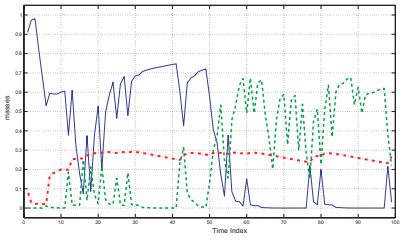


Fig. 11. Filtered θ_1, θ_3 and I_t for DSmH using Florea's QA



should also observe that sometimes $\sum_i \Pr(\theta_i) \geq 1$ because we must take into account the pignistic probability given to intersections. Note that, as explained in the introduction, our case requires such basic belief assignment. However, for cases requiring empty intersection, contraints could be added to the process. As it is required by the GPT decision process, we should take the object with the most significant amount of pignistic probability to make the decision at time τ. We can see from the figure (7) that the decision would be taken correctly through all the simulation.

B. Cases using the DSmH combining rule

1) Decision case based on the DSmH: Figures (8, 9) shows masses of four objects for the DSmH combining rule case. The first figure shows $m(\theta_1)$ and $m(\theta_3)$ for which there is little activity with $m(\theta_1)$ being higher than $m(\theta_3)$ for the first half of the simulation, both having little confidence. On the other hand, the second figure, which shows us $m(\theta_1 \cap \theta_2)$, (the solid-line), and $m(\theta_2 \cap \theta_3)$, (the dashed-line), accumulates most of the mass. $m(\theta_1 \cap \theta_2)$ is dominant in the first half of the simulation. $m(\theta_1 \cap \theta_2)$ and $m(\theta_2 \cap \theta_3)$ represents "assumed friend" and "suspect" allegiance state respectively. As we can see, the DSmH has the same problem as the conjunctive rule: accumulating the mass on conjunctions.

2) Decision case based on sum of masses for DSmH: Figure (10) shows two curves, both represents sum of masses including θ_i such as the sum $m(\theta_i) + m(\theta_i \cap \theta_j) + m(\theta_i \cap \theta_k) + m(\theta_i \cup \theta_j) + m(\theta_i \cup \theta_k)$ for $i, j, k \in \{1, 2, 3\}$ where $i \neq j \neq k$.

Figure (10) shows that we can make decisions bases on DSmH's results after a few simple arithmetics. If we use a threshold of 70% before making a decision, we can make the correct decision most of the time. It should be noted that procceding this way gives the curves of figure (10) which ressembles the DS seen on figure (6). Using curves based on summations using fewer objects lessens the decision's efficiency and increases the oscillation behavior of the output after combination.

3) Decision based on Florea's QA: Figure (11) gives the filtered values for the mass given to θ_1, θ_3 and I_t for a DSmH combination using Florea's QA. Filtering the output was necessary since the unfiltered version was too unstable with an excessive oscillation preventing us from making any decision.

In figure (11), we're able to take a decision. However, the confidence is lower than expected. Except a few times, it never exceeds 70%. It should also be observed that with this combining method, we obtained a high level of $m(I_t)$ as showed by the dash-dot line around mass of 30%. This effect hasn't been observed with the QA version using the DS rule shown in the following section. Further research would be required to investigate the source of the problem.

4) Decision based on Florea's QA using DS rule: The modified version of QA using the DS combining rule presented in figure (12) has the same behavior as the DS combining rule with an increased oscillatory behavior. The solid-line

Fig. 12. Filtered θ_1, θ_3 and I_t for DSmH with Florea's QA using DS rule

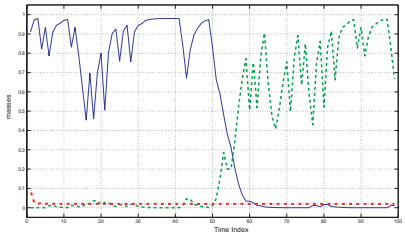
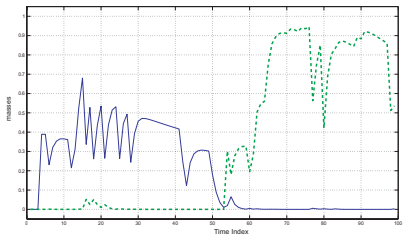


Fig. 13. $m(\theta_1 \cap \theta_2)$ and $m(\theta_2 \cap \theta_3)$ for DSmH with filtered ESM



represents $m(\theta_1)$, $m(\theta_3)$ being the dashed one. The dash-dot line near 0.02 represents the mass given to total ignorance $m(I_t)$.

5) Decision based on filtered ESM input: Figure (13) should not be considered alone but in comparison with figure (9) while looking at the ESM inputs shown in figure (5). We can see that the more an event is isolated, the less that event's occurrence has an impact on the combining rule. As described in section (II-F), the filter has the effect of reducing the mass of locally rare events.

IV. Conclusions

As introduced earlier, our research covered the use of the considered too complex Dezert-Smarandache Hybrid combining rule for the evaluation of the state of allegiance of detected targets by electronic support measures (ESM). We have compared the behavior of the DSmT in such context with the Dempster-Shafer rule and with different versions of combination rules. We have explored the use of two types of quasi-associativity algorithm (Florea's QA [9], Florea's QA with DS rule), two types of filtering methods (on the combining output's masses, on ESM's bba) and three types of decisions (with masses, with sum of masses, using generalized pignistic transformation).

A. Results review

1) Using the Dempster-Shafer combining rule: As we have seen the DS combining rule enables us to take directly from the output mass an efficient and quick decision. By quick, we mean that it is easy to modify the decision quickly after a change of allegiance. The rule is also resistant to random errors in detection keeping the decision right as it should. When we base our decision on the pignistic probability function the decision is taken correctly even with high level of pignistic probability given to some objects.

2) Using the Dezert-Smarandache hybrid rule: With a threshold of 70%, we are unable to make a decision based on bba. In rare cases, we can do so with intersections, however, even in those cases, the confidence isn't very high. However, when we base the decision on sum of masses as described in section (III-B.2), we are able to make a good decision. Since summed masses with this method adds masses from different objects, we are unable to choose precisely one of them.

3) DSmH using Florea's QA: As in the non-QA version of the DSmH rule, we are faced with a level of confidence not high enough to make a decision based on a threshold of 70%. If we consider a lower threshold, we are able to make the right decision. Note that even filtered, the output has quite an oscillatory behavior. A stronger filter might be recommanded. However, we must consider the side effects of such filters, which would slow down the reaction time.

4) DSmH using Florea's QA with DS rule: This version of the Florea's QA uses the DS combining rule instead of the conjunctive rule. As a consequence, the modified DSmH behaves quite like the DS rule with little oscillation of the output. A filter at the output is already in use, a stronger one could be tried. This version of DSmH behaves quite well, making the right decision at the right time and changing it's decision in 10 iterations from a change of allegiance.

5) DSmH using a filter on ESM's bba: In the cases where we used a filter on ESM's basic belief assignment, we've observed a better reaction to locally rare singletons occurrences as it was expected. This filter attenuates the effect of events that occurs rarely in a fixed-size filtering window. Further research should be done to find out how well and how much such filter can help.

B. Encountered problems to investigate

1) With the DSm hybrid rule: The DSmH seems unable to concentrate a wide proportion of the mass to a specific object. Most of the time, it has divided the confidence in too many objects to get a high enough confidence on a single one. To palliate to this problem, we've tried to add the mass of different objects as described earlier, in section (III-B.2). However, this method has the problem of removing the advantage of using the DSmH rule, which was to get a clearer assessment on a target's allegiance. A possible method to investigate would be to lower the decision threshold when using the DSmH.

2) With DSmH using Florea's QA: As said earlier, the DSmH case using Florea's QA doesn't give high confidence to θ_1 nor θ_3. A careful look at figure (11), shows that a great level of mass is given to ignorance. We haven't found errors or mistakes in the implementation of the rule, so we may have to investigate further the rule's behavior to find the cause.

C. Possible future works on the DSmT

Our work did not cover comparative analysis of the complexity of different methodology, nor the possibility of using different decision process than the use of the GPT. Work should be done in this area to be able to completely assess the efficiency of a complete system.

There are two main possible avenues to take for future works on the topic. One is the possibility of comparing DS and DSmH to other combining rules in the same study context. There is also the possibility of re-evaluating different versions of Florea's QA, different types of filters or the presented ones but with different parameters.

More work could be done at the decision level to evaluate the rate of successful decisions. Obviously, we should also explore different simulation runs, thus trying different distributions of bba.

ACKNOWLEDGMENTS

Defense R&D Canada for their financial and technical support. Mr. Djiknavorian would also like to thank Mr. Florea for the discussions they had about Quasi-Associativity.

REFERENCES

[1] F. Smarandache and J. Dezert (Editors), *Advances and Applications of DSmT for Information Fusion*, (Collected Works), vol. 1, USA : American Research Press, Rehoboth, 2004.

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

[3] L. Zadeh, "On the validity of Dempster's rule of combination," *Memo M 79/24*, Univ. of California, Berkeley, 1979.

[4] F. Smarandache, "Unification of Fusion Theories," *ArXiv Computer Science e-prints*, arXiv: cs/ 0409040, 2004.

[5] P. Djiknavorian and D. Grenier, "Reducing DSmT hybrid rule complexity through optimisation of the calculation algorithm," Chapter in [6], 2006.

[6] F. Smarandache and J. Dezert (Editors), *Advances and Applications of DSmT for Information Fusion*, (Collected Works), vol. 2, USA : American Research Press, Rehoboth, 2006.

[7] Ph. Smets and R. Kennes, "The transferable belief model," *Artificial Intelligence*, Vol. 66, pp 191-234, 1994

[8] Ph. Smets, "Belief functions: the disjunctive rule of combination and the generalized Bayesian theorem," *International Journal of Approximate reasoning*, Vol. 9, pp 1-35, 1993

[9] M. C. Florea, D. Grenier and A. L. Jousselme, "Adaptive combination rule for the evidence theory," *Rapport Annuel d'Activités - Laboratoire de Radiocommunications et de Traitement du Signal*, Université Laval, pp 143 - 150, 2006

[10] I. Guttman and S.S. Wilks, *Introductory Engineering Statistics*, USA : John Wiley & Sons, Inc., Second Printing, 1967.

Annexe C

Publication 2

Chapitre 15 du livre [26] publié en 2006.

Reducing DSmT hybrid rule complexity throught optimisation of the calculation algorithm

Pascal Djiknavorian
pascal.djiknavorian.1@ulaval.ca

Dominic Grenier
dominic.grenier@gel.ulaval.ca

Département de Génie Électrique et de Génie Informatique
Faculté des Sciences et de Génie, Université Laval
Québec, Canada, G1K 7P4

Abstract - ***The work covered here had for objective to write a Matlab program able to execute efficiently, the DSmT hybrid rule of combination. As we know, the DSmT hybrid rule of combination is highly complex to execute and requires high amounts of ressources. We have introduced a novel way of understanding and treating the rule of combination and thus were able to develop a Matlab program that would avoid the high level of complexity and ressources needs.***

Keywords: DSmT hybrid rule, optimisation, complexity, Matlab.

1 Introduction

The purpose of DSmT [1] was to introduce a theory that would allow to correctly fuse data, even in presence of conflicts between sources of evidence or in presence of constraints. However, as we know, the DSmT hybrid rule of combination is very complex to compute and use in data fusion compared to other rules of combination [2]. We will show in the following sections, that there's a way to avoid the high level of complexity of DSmT hybrid rule of combination permitting to program it into Matlab code. We will begin by a brief review of the theory used in the subsequent sections, where will be presented few definitions followed by a review of Dempster-Shafer theory and it's problem with mass redistribution. We will then look at Dezert-Smarandache theory and it's complexity. It is followed by a section presenting the methodology used to avoid the complexity of DSmT hybrid rule of combination. We will conclude with a short performance analysis and with the developped Matlab code in appendix.

2 Theories

2.1 Definitions

A minimum of knnowledge is required to understand DSmT, we'll thus begin with a short review of important concepts.

- *Frame of discernment* (Θ) : $\Theta = \{\theta_1, \theta_2, \dots \theta_n\}$. It's the set including every possible objects θ_i.
- *Power set* (2^Θ): represents the set of all possible sets using the objects of the frame of discernment Θ. It includes the empty set and excludes intersections. With the frame of discernment we've defined above, we get the power set $2^\Theta = \{\emptyset, \{\theta_1\}, \{\theta_2\}, \dots \{\theta_n\}, \{\theta_1, \theta_2\}, \dots \{\theta_1, \theta_2, \dots \theta_n\}, \dots, \Theta\}$.
- *Hyperpower set* (D^Θ): represents the set of all possible sets using the objects of the frame of discernment Θ. It includes the empty set. With the frame of discernment $\Theta = \{\theta_1, \theta_2\}$, we get the hyperpower set $D^\Theta = \{\emptyset, \{\theta_1\}, \{\theta_2\}, \{\theta_1 \cap \theta_2\}, \{\theta_1 \cup \theta_2\}\}$.
- *Belief* ($\mathrm{Bel}(A)$): is an evaluation of the minimal level of certainty, or trust, that a set can have.
- *Plausibility* ($\mathrm{Pl}(A)$): is an evaluation of the maximal level of certainty, or trust, that a set can have.
- *Constraint* : a set considered impossible to obtain.
- *Basic belief assignement* (bba) : $m : 2^\Theta \to [0, 1]$, so the mass given to a set $A \subseteq \Theta$ follows $m(A) \in [0, 1]$.
- *Core of* Θ ($\mathcal{K}$): The set of all focal elements of Θ, where a focal element is a subset A of Θ such that $m(A) > 0$.

2.2 Dempster-Shafer Theory

The DST rule of combination is a conjunctive normalized rule working on the power set as described previously. It combines information with intersections, meaning that it works only with the bba's intersections. The theory also makes the hypothesis that the sources of evidence are mathematically independent. The i^{th} bba's source of evidence is denoted m_i. Equation (1) describes the DST rule of combination where K is the conflict. The conflict in DST is defined as in equation (2).

$$(m_1 \oplus m_2)(C) = \frac{1}{1-K} \sum_{A \cap B = C} m_1(A)\, m_2(B) \qquad \forall C \subseteq \Theta \tag{1}$$

$$K = \sum_{A \cap B = \emptyset} m_1(A)\, m_2(B) \qquad A, B \subseteq \Theta \tag{2}$$

2.2.1 DST combination example

Let's consider the case where we have an air traffic survaillance officer in charge of monitoring readings from two radars. The radars constitutes our two sources of evidence. In this case, both radars displays a target with the level of confidence (bba) of it's probable identity. Radar 1 shows that it would be an F-16 aircraft (θ_1) with $m_1(\theta_1) = 0.50$, an F-18 aircraft (θ_2) with $m_1(\theta_2) = 0.10$, one of both with $m_1(\theta_1 \cup \theta_2) = 0.30$, or it might be another airplane (θ_3) with $m_1(\theta_3) = 0.10$. Collected data from radar 1 and 2 are shown in table 1. We can easily see from that table that the frame of discernment $\Theta = \{\theta_1, \theta_2, \theta_3\}$ is sufficient to describe this case.

The evident contradiction between the sources causes a conflict to be resolved before interpreting the results. Considering the fact that the DST doesn't admit intersections, we'll have to discard some possible sets. Also, the air traffic surveillance officer got intelligence information recommending exclusion of the case $\{\theta_3\}$, creating a constraint on $\{\theta_3\}$. Table 2 represent the first step of the calculation before the redistribution of the conflicting mass.

Table 1: Events from two sources of evidence to combine

(2^Θ)	$m_1(A)$	$m_2(A)$
$\{\theta_1\}$	0.5	0.1
$\{\theta_2\}$	0.1	0.6
$\{\theta_3\}$	0.1	0.2
$\{\theta_1, \theta_2\}$	0.3	0.1

Table 2: Results from disjunctive combination of information from table 1 before mass redistribution

	$m_1(\theta_1)$ 0.5	$m_1(\theta_2)$ 0.1	$m_1(\theta_3)$ 0.1	$m_1(\theta_1 \cup \theta_2)$ 0.3
$m_2(\theta_1)$ 0.1	θ_1 0.05	$\theta_1 \cap \theta_2 = \emptyset$ 0.01	$\theta_1 \cap \theta_3 = \emptyset$ 0.01	θ_1 0.03
$m_2(\theta_2)$ 0.6	$\theta_1 \cap \theta_2 = \emptyset$ 0.30	θ_2 0.06	$\theta_2 \cap \theta_3 = \emptyset$ 0.06	θ_2 0.18
$m_2(\theta_3)$ 0.2	$\theta_1 \cap \theta_3 = \emptyset$ 0.10	$\theta_2 \cap \theta_3 = \emptyset$ 0.02	$\theta_3 = \emptyset$ 0.02	$(\theta_1 \cup \theta_2) \cap \theta_3 = \emptyset$ 0.06
$m_2(\theta_1 \cup \theta_2)$ 0.1	θ_1 0.05	θ_2 0.01	$(\theta_1 \cup \theta_2) \cap \theta_3 = \emptyset$ 0.01	$\theta_1 \cup \theta_2$ 0.03

As we can see in table 2, the total mass of conflict is $\sum m(\emptyset) = 0.59$. So among all the possible sets, 0.59 of the mass is given to $\emptyset$. Which would make it the most probable set. Using equation (1) the conflict is redistributed proportionally among non empty sets. Results are given in tables 3 and 4. Finally, we can see that the most probable target identity is an F-18 aircraft.

Table 3: Results from disjunctive combination of information from table 1 with mass redistribution

	$m_1(\theta_1)$ 0.5	$m_1(\theta_2)$ 0.1	$m_1(\theta_3)$ 0.1	$m_1(\theta_1 \cup \theta_2)$ 0.3
$m_2(\theta_1)$ 0.1	θ_1 $\frac{0.05}{1-0.59}$	$\emptyset$ 0	$\emptyset$ 0	θ_1 $\frac{0.03}{1-0.59}$
$m_2(\theta_2)$ 0.6	$\emptyset$ 0	θ_2 $\frac{0.06}{1-0.59}$	$\emptyset$ 0	θ_2 $\frac{0.18}{1-0.59}$
$m_2(\theta_3)$ 0.2	$\emptyset$ 0	$\emptyset$ 0	$\emptyset$ 0	$\emptyset$ 0
$m_2(\theta_1 \cup \theta_2)$ 0.1	θ_1 $\frac{0.05}{1-0.59}$	θ_2 $\frac{0.01}{1-0.59}$	$\emptyset$ 0	$\theta_1 \cup \theta_2$ $\frac{0.03}{1-0.59}$

Table 4: Final results for the example of DST combination

$m_{1\oplus 2}(\emptyset)$	0.000
$m_{1\oplus 2}(\theta_1)$	0.317
$m_{1\oplus 2}(\theta_2)$	0.610
$m_{1\oplus 2}(\theta_1 \cup \theta_2)$	0.073

The problem, which was predictable by analytical analysis of equation (1), occurs when conflict (K) get closer to 1. As K grows closer to 1, the DST rule of combination tends to give incoherent results.

2.3 Dezert-Smarandache Theory

Instead of the power set, used in DST, the DSmT uses the hyperpower set. DSmT is thus able to work with intersections. They also differ by their rules of combination. DSmT posseses two rules of combination which are able to work around the conflicted mass redistribution problem:

- Classic DSm rule of combination (DSmC), which is based on the free model $M^f(\Theta)$

$$m(C) = \sum_{A \cap B = C} m_1(A)\, m_2(B) \qquad A, B \in D^\Theta, \forall C \in D^\Theta \tag{3}$$

- Hybrid DSm rule of combination (DSmH), which is able to work with many types of constraints

$$m_{M(\Theta)}(A) = \phi(A)\left[S_1(A) + S_2(A) + S_3(A)\right] \tag{4}$$

$$S_1(A) = \sum_{X_1 \cap X_2 = A} m_1(X_1)\, m_2(X_2) \qquad \forall X_1, X_2 \in D^\Theta \tag{5}$$

$$S_2(A) = \sum_{[(u(X_1) \cup u(X_2)) = A] \vee [[(u(X_1) \cup u(X_2)) \in \emptyset) \wedge (A = I_t)]} m_1(X_1)\, m_2(X_2) \qquad \forall X_1, X_2 \in \emptyset \tag{6}$$

$$S_3(A) = \sum_{X_1 \cup X_2 = A} m_1(X_1)\, m_2(X_2) \qquad \forall X_1, X_2 \in D^\Theta \quad \text{and} \quad X_1 \cap X_2 \in \emptyset \tag{7}$$

Note that $\phi(A)$ in equation (4) is a binary fonction resulting in 0 for empty or impossible sets and in 1 for non empty sets. In equation (6), $u(X)$ represents the union of all objects of set X. Carefull analysis of equation (7) tells us that it's the union of all objects of sets X_1 and X_2, when it is not empty. Finally, also from equation (6), I_t represents the total ignorance, or the union of all objects part of the frame of discernment. Futher information on how to understand and proceed in the calculation of DSmH is available in subsequent sections.

2.3.1 DSmC combination example

This example cannot be resolved by DST because of highly conflictual sources of evidence (K tends toward 1). Sources' information shown in table 5 gives us, with DSmC, the results displayed in table 6. As we can see, no mass is associated to an empty set since DSmC doesn't allow constraints. Final results for the present example, given by table 7, tells us that the most probable identity of the target to identify is an hybrid of objects of type θ_1 and θ_2.

Table 5: Events from two sources of evidence to combine

(D^Θ)	$m_1(A)$	$m_2(A)$
$\{\theta_1\}$	0.8	0.0
$\{\theta_2\}$	0.0	0.9
$\{\theta_3\}$	0.2	0.1
$\{\theta_1, \theta_2\}$	0.0	0.0

Table 6: Results from DSmC rule of combination with table 1 informations

	$m_1(\theta_1)$ 0.8	$m_1(\theta_2)$ 0.0	$m_1(\theta_3)$ 0.2	$m_1(\theta_1 \cup \theta_2)$ 0.0
$m_2(\theta_1)$ 0.0	θ_1 0.00	$\theta_1 \cap \theta_2$ 0.00	$\theta_1 \cap \theta_3$ 0.00	θ_1 0.00
$m_2(\theta_2)$ 0.9	$\theta_1 \cap \theta_2$ 0.72	θ_2 0.00	$\theta_1 \cap \theta_3$ 0.18	θ_2 0.00
$m_2(\theta_3)$ 0.1	$\theta_1 \cap \theta_3$ 0.08	$\theta_2 \cap \theta_3$ 0.00	θ_3 0.02	$(\theta_1 \cup \theta_2) \cap \theta_3$ 0.00
$m_2(\theta_1 \cup \theta_2)$ 0.0	θ_1 0.00	θ_2 0.00	$(\theta_1 \cup \theta_2) \cap \theta_3$ 0.00	$\theta_1 \cup \theta_2$ 0.00

Table 7: Final results for the example of DSmC combination ($m_{1\oplus 2}$)

(θ_1)	0.00	$(\theta_1 \cap \theta_2)$	0.72	$(\theta_1 \cup \theta_2)$	0.00
(θ_2)	0.00	$(\theta_1 \cap \theta_3)$	0.08	$(\theta_1 \cup \theta_3)$	0.00
(θ_3)	0.02	$(\theta_2 \cap \theta_3)$	0.18	$(\theta_2 \cup \theta_3)$	0.00
$(\emptyset)$	0.00	$(\theta_1 \cap \theta_2) \cup \theta_3$	0.00	$(\theta_1 \cup \theta_2) \cap \theta_3$	0.00
$(\theta_1 \cup \theta_2 \cup \theta_3)$	0.00	$(\theta_1 \cap \theta_3) \cup \theta_2$	0.00	$(\theta_1 \cup \theta_3) \cap \theta_2$	0.00
$(\theta_1 \cap \theta_2 \cap \theta_3)$	0.00	$(\theta_2 \cap \theta_3) \cup \theta_1$	0.00	$(\theta_2 \cup \theta_3) \cap \theta_1$	0.00

3 How to avoid the complexity

3.1 Simpler way to view the DSmT hybrid rule of combination

First of all, one simple thing to do to keep the use of ressources at low levels is to keep only the usefull data. For example, table 6 shouldn't be entered as is in a program but reduced to the equivalent table 8. This way, the only allocated space to execute the calculation is the data space we actually need and use.

This is specially important for full explicit calculation of equation (4), as the number of possible objects and/or the number of possible sources of evidence grows, we would hence avoid extraordinary increase in ressources needs. Hence avoiding Dedekind's sequence progression in needs. [1]

3.1.1 Simple procedure for effective DSmH

Instead of viewing DSmH as a mathematical equation, we propose to view it as a procedure. Table 9 displays that procedure. Obviously, it is still equivalent to the mathematical equation, but this way has the advantage of being very easily understood and implemented. The ease in implementation is due to the high ressemblance of the procedure to pseudo-code, a common step in the software design process.

Table 8: Reduced version of table 6

	$m_1(\theta_1)$ 0.8	$m_1(\theta_3)$ 0.2
$m_2(\theta_2)$ 0.9	$\theta_1 \cap \theta_2$ 0.72	$\theta_1 \cap \theta_3$ 0.18
$m_2(\theta_3)$ 0.1	$\theta_1 \cap \theta_3$ 0.08	θ_3 0.02

Table 9: Procedure to apply to each pairs of sets (X_1, X_2) until it's combined mass is given to a set

Step S1 $(\theta_1 \cap \theta_2)$	If $(\theta_1 \cap \theta_2)$ is a constraint, then continue at step S3, otherwise, the mass $m_1(X_1)\, m_2(X_2)$ is added to the mass $A = (\theta_1 \cap \theta_2)$.
Step S3 $(\theta_1 \cup \theta_2)$	If $(\theta_1 \cup \theta_2)$ is a constraint, then continue at step S2, otherwise, the mass $m_1(X_1)\, m_2(X_2)$ is added to the mass $A = (\theta_1 \cup \theta_2)$.
Step S2 $(u(X_1) \cup u(X_2))$	If $(u(X_1) \cup u(X_2))$ is a constraint, then add mass to I_t, otherwise, the mass $m_1(X_1)\, m_2(X_2)$ is added to the mass $A = (u(X_1) \cup u(X_2))$.

3.2 Notation system used

3.2.1 Sum of products

The system we conceived treats information in terms of *union of intersections* or *sum of products*. The sum (ADD) being represented by union ($\cup$), and the product (MULT) by intersection ($\cap$). We have choosen this, instead of *product of sums* to avoid having to treat parenthesis. We could also use the principles developped for logic circuits as Karnaugh table, boolean rules, etc. Here are few examples of this notation:

- $\theta_1 \cap \theta_2 \cap \theta_3 = \theta_1\theta_2\theta_3 = [1, MULT, 2, MULT, 3]$
- $\theta_1 \cup \theta_2 \cup \theta_3 = \theta_1 + \theta_2 + \theta_3 = [1, ADD, 2, ADD, 3]$
- $(\theta_1 \cap \theta_2) \cup \theta_3 = \theta_1\theta_2 + \theta_3 = [1, MULT, 2, ADD, 3] = [3, ADD, 1, MULT, 2]$
- $(\theta_1 \cup \theta_2) \cap \theta_3 = (\theta_1 \cap \theta_3) \cup (\theta_2 \cap \theta_3) = \theta_1\theta_3 + \theta_2\theta_3 = [1, MULT, 3, ADD, 2, MULT, 3]$
- $(\theta_1 \cap \theta_2 \cap \theta_3) \cup (\theta_4 \cap \theta_5) = \theta_1\theta_2\theta_3 + \theta_4\theta_5 = [1, MULT, 2, MULT, 3, ADD, 4, MULT, 5]$

3.2.2 Conversion between *sum of products* and *product of sums* notation

As we've seen above, we will use the *sum of products* as our main way of writting sets. However, as we will see later, we will need to use the *product of sums* or intersections of unions in some parts of our system to simplify the calculation process. More specifically, this dual system of notation, introduced in the two last columns of table 10, was done so we would be able to use the same algorithm to work with a matrix of unions and a matrix of intersections. Table 10 thus presents the notation used in the following, accompanied with it's equivalent mathematical notation. We can see in the *sum of products* notation in table 10, that a line represents a monome of product type (eg. $\theta_1\theta_3$) and that lines are then summed to get unions (eg. $\theta_1\theta_3 + \theta_2$). In the *product of sums* notation, we have the reversed situation where lines represents a monome of sum type (eg. $\theta_1 + \theta_3$) and that lines are then multiplied to get intersections (eg. $\theta_2(\theta_1 + \theta_3)$).

The difficult part is the conversion step from the *sum of products* to the *product of sums* notation. For the simple cases, such as the ones presented in the first three lines of table 10 consist only in changing matrices lines into columns and columns into lines. For simplification in the conversion process we also use the absorption rule as described in equation (8) which is derived from the fact that $(\theta_1\theta_2) \subseteq \theta_1$. Using that rule, we can see how came the two last rows of table 10 by looking at the process detailed in equations (9) and (10).

$$\theta_1 + \theta_1\theta_2 = \theta_1 \tag{8}$$

$$\begin{aligned}(\theta_1 \cup \theta_2) \cap (\theta_2 \cup \theta_3) = (\theta_1 + \theta_2)(\theta_2 + \theta_3) &= \theta_1\theta_2 + \theta_1\theta_3 + \theta_2 + \theta_2\theta_3 \\ &= \theta_1\theta_3 + \theta_2\end{aligned} \tag{9}$$

$$(\theta_1 \cap \theta_2) \cup (\theta_2 \cap \theta_3) = \theta_1\theta_2 + \theta_2\theta_3 = \theta_2(\theta_1 + \theta_3) \tag{10}$$

Table 10: Equivalent notations for events

Mathematical	Matlab input/output	Sum of products	Product of sums
$\{\theta_1\}$	$[1]$	$\begin{bmatrix} 1 \end{bmatrix}$	$\begin{bmatrix} 1 \end{bmatrix}$
$\{\theta_1 \cup \theta_2\}$	$[1, ADD, 2]$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \end{bmatrix}$
$\{\theta_1 \cap \theta_2\}$	$[1, MULT, 2]$	$\begin{bmatrix} 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$
$\{(\theta_2) \cup (\theta_1 \cap \theta_3)\}$	$[2, ADD, 1, MULT, 3]$	$\begin{bmatrix} 2 \\ 1 & 3 \end{bmatrix}$	–
$\{(\theta_1 \cup \theta_2) \cap (\theta_2 \cup \theta_3)\}$	–	–	$\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$
$\{(\theta_1 \cap \theta_2) \cup (\theta_2 \cap \theta_3)\}$	$[1, MULT, 2, ADD, 2, MULT, 3]$	$\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$	–
$\{(\theta_2) \cap (\theta_1 \cup \theta_3)\}$	–	–	$\begin{bmatrix} 2 \\ 1 & 3 \end{bmatrix}$

However, in the programmed Matlab code, the following procedure is used and works for any case. It's based on the use of DeMorgan's laws as seen in equations (11) and (12). Going through DeMorgan twice lets us avoid the use of negative sets. Hence, we will still respect DSm Theory even with the use of this mathematical law. The use of absorption rule, as described in equation (8) also helps us acheive better simplification.

$$\overline{A}\,\overline{B} = \overline{A \,+\, B} \tag{11}$$

$$\overline{A} \,+\, \overline{B} = \overline{A\,B} \tag{12}$$

Here's how we proceed for the case of conversion from a set in *sum of products* to a set in *product of sums* notation. It's quite simple actually, we begin with an inversion of operators (changing additions ($\cup$) for multiplications ($\cap$) and multiplications for additions), followed by distribution of products and a simplification step. We then end it with a second inversion of operators. Since we've used the inversion two times, we don't have to indicate the not operator, ($\overline{\overline{A}} = A$).

Let's now proceed with a short example. Suppose we want to convert the set shown in equation (13) to a set in product of sums notation. We proceed first as said, with an inversion of operators resulting in the set in equation (14). We then distribute the multiplication as we did to get the set in equation (15). This is then followed by a simplification giving us equation (16) and a final inversion of operators gives us the set in equation (17). The set in equation (17) represents the product of sums notation version of the set in equation (13), which is in sum of products. A simple distribution of products and simplification can get us back from (17) to (13).

$$\theta_1 + \theta_2\theta_3 + \theta_2\theta_4 \tag{13}$$

$$\overline{\overline{(\theta_1)}\ \left(\overline{\theta_2} + \overline{\theta_3}\right)\ \left(\overline{\theta_2} + \overline{\theta_4}\right)} \tag{14}$$

$$\overline{\overline{\theta_1}\,\overline{\theta_2} + \overline{\theta_1}\,\overline{\theta_2}\,\overline{\theta_4} + \overline{\theta_1}\,\overline{\theta_2}\,\overline{\theta_3} + \overline{\theta_1}\,\overline{\theta_3}\,\overline{\theta_4}} \tag{15}$$

$$\overline{\overline{\theta_1}\,\overline{\theta_2} + \overline{\theta_1}\,\overline{\theta_3}\,\overline{\theta_4}} \tag{16}$$

$$(\theta_1 + \theta_2)\,(\theta_1 + \theta_3 + \theta_4) \tag{17}$$

3.3 How simple can it be

We have completed conception of a Matlab code for the dynamic case. We've tried to optimize the code but some work is still necessary. It's now operationnal for restraint body of evidence and well behaving. Here's an example of the input required by the system with the events from table 11. We will also proceed with θ_2 as a constraint making the following constraints too:

- $\theta_1 \cap \theta_2 \cap \theta_3$

- $\theta_1 \cap \theta_2$
- $\theta_2 \cap \theta_3$
- $(\theta_1 \cup \theta_3) \cap \theta_2$

Note that having θ_2 as a constraint, has an impact on more cases than the enumarated ones above. In fact, if we obtain cases like $\theta_1 \cup \theta_2$ for instance, since θ_2 is a constraint, the resulting case would then be θ_1. We will have to consider this when evaluating final bba for the result.

Table 11: Information from three sources

(D^Θ)	$m_1(A)$	$m_2(A)$	$m_3(A)$
$\{\theta_1\}$	0.7	0.0	0.1
$\{\theta_2\}$	0.0	0.6	0.1
$\{\theta_3\}$	0.2	0.0	0.5
$\{\theta_1 \cup \theta_2\}$	0.0	0.0	0.3
$\{\theta_1 \cup \theta_3\}$	0.0	0.2	0.0
$\{\theta_2 \cup \theta_3\}$	0.0	0.2	0.0
$\{\theta_2 \cap \theta_3\}$	0.1	0.0	0.0

As we can see, only the non zero values are given as input to the system. Which gives us also only non zero values. Moreover, the input elements are all given in *sum of products* only. The output of the system is also given in *sum of products*. The output also gives us results for Belief and Plausibility for the case given as input.

Notice also that we have dynamic constraints capability, meaning that we can put constraints on each steps of combination. They can also differ at each steps of the calculation. Instead of considering constraints only at the final step of combination, this system is thus able to reproduce real fusion condition where constraints may vary. Three different cases are presented here, keeping the same input information but varying the constraints conditions.

Complete commented listing of the produced Matlab code is available in the appendix. For the present section, only the parameters required in input and the output are displayed.

```
% Example with dynamic constraints kept stable
% INPUT FOR THE MATLAB PROGRAM

number_sources    = 3;   kind    = ['dynamic'];
info(1).elements = {[1],[3], [2, MULT, 3]};  info(1).masses   = [0.7, 0.2, 0.1];
info(2).elements = {[2],[1, ADD, 3], [2, ADD, 3]};  info(2).masses   = [0.6, 0.2, 0.2];
info(3).elements = {[1], [2], [3], [1, ADD, 2]};  info(3).masses   = [0.1, 0.1, 0.5, 0.3];
constraint{1}    = {[2], [1, MULT, 2], [2, MULT, 3],...
                    [1, MULT, 2, ADD, 3, MULT, 2], [1, MULT, 2, MULT, 3]};
constraint{2}    = {[2], [1, MULT, 2], [2, MULT, 3],...
                    [1, MULT, 2, ADD, 2, MULT, 3], [1, MULT, 2, MULT, 3]};

% OUTPUT OF THE MATLAB PROGRAM

DSm hybride                  Plausibility              Belief
1 : m=0.28800000             1 : m=1.00000000          1 : m=0.82000000
1 MULT 3 : m=0.53200000      1 MULT 3 : m=1.00000000   1 MULT 3 : m=0.53200000
3 : m=0.17800000             3 : m=1.00000000          3 : m=0.71000000
1 ADD 3 : m=0.00200000       1 ADD 3 : m=1.00000000    1 ADD 3 : m=1.00000000

% Example with dynamic constraints applied only once at the end
% CONSTRAINTS INPUT FOR THE MATLAB PROGRAM

constraint{1}    = {};
constraint{2}    = {[2], [1, MULT, 2], [2, MULT, 3],...
                    [1, MULT, 2, ADD, 2, MULT, 3], [1, MULT, 2, MULT, 3]};

% OUTPUT OF THE MATLAB PROGRAM
```

```
DSm hybride                  Plausibility              Belief
1 : m=0.36800000             1 : m=1.00000000          1 : m=0.61000000
1 MULT 3 : m=0.24200000      1 MULT 3 : m=1.00000000   1 MULT 3 : m=0.24200000
3 : m=0.39000000             3 : m=1.00000000          3 : m=0.63200000

% Example with dynamic constraints varying between steps of calculation
% CONSTRAINTS INPUT FOR THE MATLAB PROGRAM

constraint{1}    = {[2, MULT, 3], [2, ADD, 3]};
constraint{2}    = {[2], [1, MULT, 2], [2, MULT, 3],...
                    [1, MULT, 2, ADD, 2, MULT, 3], [1, MULT, 2, MULT, 3]};

% OUTPUT OF THE MATLAB PROGRAM

DSm hybride                  Plausibility              Belief
1 : m=0.31200000             1 : m=1.00000000          1 : m=0.55400000
1 ADD 3 : m=0.14800000       1 ADD 3 : m=1.00000000    1 ADD 3 : m=1.00000000
1 MULT 3 : m=0.24200000      1 MULT 3 : m=1.00000000   1 MULT 3 : m=0.24200000
3 : m=0.29800000             3 : m=1.00000000          3 : m=0.54000000
```

3.4 Optimization in the calculation algorithm

3.4.1 How does it work

Being treated by a vectorial interpreater, our Matlab code had to be adapted in consequence. We have also been avoiding, as much as we could, the use of `for` and `while` loops.

Our Matlab code was conceived with two main matrices, one containing intersections, the other one containing unions. The input information is placed into a matrix identified as the fusion matrix. When building this matrix, our program puts in a vector each unique objects that will be used, hence defining total ignorance (I_t) for the case in input. Each elements of this matrix is a structure having two fields: sets and mass. Note also that only the first row and column of the matrix is filled with the input information. The rest of the matrix will contain the result.

It is easier to proceed with the intersection between two sets A and B using *product of sums* and to proceed with the union $A \cup B$ using *sum of products*. Because of that we've choosen to keep the intersection matrix in the *product of sums* notation and the union matrix in the *sum of products* while working on these matrices separately.

To build the union matrix, we use information from the fusion matrix with the *sum of products* notation. The intersection matrix uses the *product of sums* notation for it's construction with the information from the fusion matrix. However, once the intersection matrix is built, a simple conversion to the *sum of products* notation is done as we've described earlier. This way, data from this table can be compatible with those from the fusion and the union matrices.

Once the basis of the union matrix is defined, a calculation of the content is done by evaluating the result of the union of every non-empty elements combination $m_1(X_i)\, m_2(X_j)$. The equivalent is done with the intersection matrix, replacing the union with an intersection obviously. Once the calculation of the content of the intersection matrix completed, it is converted to the sum of product notation.

The next step consist to fill up the fusion matrix with the appropriate information depending on the presence of constraints and following the procedure described earlier for the calculation of DSmH combination rule. So, if there is no constraint on the ij^{th} element of the matrix, we take the information from the intersection matrix in S_1; if it's constrained, we take the information from the union matrix in S_3; if it's constrained, we take the information from $(u(X_1) \cup u(X_2))$ in S_2; if it's constrained, we then go to the total ignorance I_t.

In the case we want to fuse information from more than two sources, we could choose to fuse the information dynamically or statically. The first case being done by fusing two sources at a time. The latter case considers information from all sources at once. Note however that our code is only able to proceed with the calculation dynamically.

We will now proceed step by step with a full example, interlaced with some explanation on the procedure, using the information from table 11 and the constraints described above that table.

Table 12: Union matrix with bba's m_1, m_2 information from table 11 in *sum of products* notation

m_1 \ m_2	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.6	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.2
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.7	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 0.42	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ 0.14
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.12	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.04	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.04
$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.06	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.02	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.02

Table 12 gives us the union result from each combination of non zero bba from the first two sources of evidence. The notation used in the case for union matrices is the sum of products. In the case of table 13, the intersection matrix, it's first built in the product of sums notation so the same calculation algorithm can be used to evaluate the intersection result from each combination of non zero bba from the first two sources of evidence as it was used in the union matrix. As we'll see, a conversion to the sum of products notation is done to be able to obtain table 14.

We obtained $\theta_2\theta_3$ as a result in the second result cell in the last row of table 13 because the intersection $(\theta_2 \cdot \theta_3) \cdot (\theta_1 + \theta_3)$ gives us $\theta_1\theta_2\theta_3 + \theta_2\theta_3$ which, following absorption rule, gives us $\theta_2\theta_3$. The same process occurs on the second result cell in the first row of the same table where $\theta_1 \cdot (\theta_1 + \theta_3) = \theta_1 + \theta_1\theta_3 = \theta_1$.

Table 13: Intersection matrix with bbas m_1, m_2 information from table 11 in *product of sums* notation

m_1 \ m_2	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.6	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.2
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.7	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 0.42	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 \\ 2 & 3 \end{bmatrix}$ 0.14
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.12	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.04	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.04
$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.06	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.02	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.02

Table 14: Intersection matrix with bbas m_1, m_2 information from table 11 in *sum of products* notation

m_1 \ m_2	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.6	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.2
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.7	$\begin{bmatrix} 1 & 2 \\ \end{bmatrix}$ 0.42	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$ 0.14
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.12	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.04	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.04
$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.06	$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.02	$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.02

From the tables 12 and 14 we proceed with the DSmH and choose, according to constraints, from which table the result will come. We might also have to evaluate $(u(X_1) \cup u(X_2))$, or give the mass to the total ignorance if the intersection and union matrices' sets are constrained. We've displayed the choice made in the fusion matrix in table 15 with these symbols $\cap$ (intersection), $\cup$ (union), **u** (union of the sum of objects of combined sets), **It** (total ignorance). As you will see, we've choosen a case where we have constraints applied at each steps of combination. (ie.: when $[m_1, m_2]$ and when $[m_1 \oplus m_2, m_3]$ are combined.) Table 16 is the simplified version of table 15 in which sets has been adapted to consider constraints. It's followed by table 17 which represents the results from the first combination.

As we can see in table 15, the first result cell from the first row was obtained from the union matrix because $\theta_1 \cap \theta_2$ is a constraint. Also, the first result cell from the last row was obtained from the union of the sum of objects of the combined sets because $\theta_2 \cap \theta_3$ is a constraint in the intersection table (table 14) at the same position, so is θ_2 in the union table (table 12).

Table 15: Fusion matrix with bbas m_1, m_2 information from table 11 in *sum of products* notation

m_1 \ m_2	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.6	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.2
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.7	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 0.42 $\cup$	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.14 $\cap$	$\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$ 0.14 $\cap$
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.12 $\cup$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.04 $\cap$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.04 $\cap$
$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.06 **u**	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.02 $\cup$	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.02 $\cup$

Table 16: Simplified fusion matrix with bbas m_1, m_2 information from table 11 in *sum of products* notation

m_1 \ m_2	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.6	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.2	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.2
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.7	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.42 $\cup$	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.14 $\cap$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.14 $\cap$
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.2	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.12 $\cup$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.04 $\cap$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.04 $\cap$
$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.06 **u**	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.02 $\cup$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.02 $\cup$

Table 17: Result of the combination of m_1 and m_2 from table 11

[1]	[3]	$[1 \cup 3]$	$[1 \cap 3]$
0.56	0.28	0.02	0.14

On the first row of table 16, the first result giving us θ_1 is obtained because $\theta_1 \cup \theta_2 = \theta_1$ when θ_2 is a constraint. The same process gave us $\theta_1 \cap \theta_3$ in the last cell of the first row. In that case, we obtained that result having $\theta_1 \cap \theta_2$ as a constraint where $\theta_1\theta_2 + \theta_1\theta_3 = \theta_1\theta_3$. Since we have more than two sources and have choosen a dynamic methodology: once the first two sources combined, we will have to proceed with a second combination. This time, we combine the results from the first combination $m_1 \oplus m_2$ with the third event from source of evidence m_3.

Table 18 represents the union matrix from second combination. Table 19 and 20 are the intersection matrix with *product of sums* and *sum of products* notation respectively.

Table 18: Union matrix with bbas $m_1 \oplus m_2, m_3$ information comes from tables 11 and 15 in *sum of products* notation

$m_1 \oplus m_2$ \ m_3	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.5	$\begin{bmatrix} 1 \\ 2 \\ \end{bmatrix}$ 0.3
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.56	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.056	$\begin{bmatrix} 1 \\ 2 \\ \end{bmatrix}$ 0.056	$\begin{bmatrix} 1 \\ 3 \\ \end{bmatrix}$ 0.28	$\begin{bmatrix} 1 \\ 2 \\ \end{bmatrix}$ 0.168
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.28	$\begin{bmatrix} 1 \\ 3 \\ \end{bmatrix}$ 0.028	$\begin{bmatrix} 2 \\ 3 \\ \end{bmatrix}$ 0.028	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 \\ 2 \\ 3 \\ \end{bmatrix}$ 0.084
$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.014	$\begin{bmatrix} 1 & 3 \\ 2 \\ \end{bmatrix}$ 0.014	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.07	$\begin{bmatrix} 1 \\ 2 \\ \end{bmatrix}$ 0.042
$\begin{bmatrix} 1 \\ 3 \\ \end{bmatrix}$ 0.02	$\begin{bmatrix} 1 \\ 3 \\ \end{bmatrix}$ 0.002	$\begin{bmatrix} 1 \\ 2 \\ 3 \\ \end{bmatrix}$ 0.002	$\begin{bmatrix} 1 \\ 3 \\ \end{bmatrix}$ 0.01	$\begin{bmatrix} 1 \\ 2 \\ 3 \\ \end{bmatrix}$ 0.006

Table 19: Intersection matrix with bbas $m_1 \oplus m_2, m_3$ information from tables 11 and 15 in *product of sums* notation

$m_1 \oplus m_2$ \ m_3	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.5	$\begin{bmatrix} 1 & 2 \\ \end{bmatrix}$ 0.3
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.56	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.056	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 0.056	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.28	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.168
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.28	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.028	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.028	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 & 2 \\ 3 \end{bmatrix}$ 0.084
$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.014	$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ 0.014	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.07	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.042
$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.02	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.002	$\begin{bmatrix} 2 \\ 1 & 3 \end{bmatrix}$ 0.002	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.01	$\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$ 0.006

Finally we get table 21 which consists of the final fusion matrix, table 22 which is a simplified version of 21, and table 23 which compiles equivalent results giving us the result of the DSmH for the information from table 11 with same constraints applied at each steps of combination.

Table 20: Intersection matrix with bbas $m_1 \oplus m_2, m_3$ information from tables 11 and 15 in *sum of products* notation

$m_1 \oplus m_2$ \ m_3	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.5	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 0.3
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.56	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.056	$\begin{bmatrix} 1 & 2 \\ \end{bmatrix}$ 0.056	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.28	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.168
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.28	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.028	$\begin{bmatrix} 2 & 3 \\ \end{bmatrix}$ 0.028	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix}$ 0.084
$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.014	$\begin{bmatrix} 1 & 2 & 3 \\ \end{bmatrix}$ 0.014	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.07	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.042
$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.02	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.002	$\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$ 0.002	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.01	$\begin{bmatrix} 1 \\ 2 & 3 \end{bmatrix}$ 0.006

.

Table 21: Fusion matrix with bbas $m_1 \oplus m_2, m_3$ information from table 11 and 15 in *sum of products* notation

$m_1 \oplus m_2$ \ m_3	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.5	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 0.3
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.56	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.056 $\cap$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 0.056 $\cup$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.28 $\cap$	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.168 $\cap$
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.28	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.028 $\cap$	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 0.028 $\cup$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.14 $\cap$	$\begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix}$ 0.084 $\cap$
$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.014 $\cap$	$\begin{bmatrix} 1 & 3 \\ 2 \end{bmatrix}$ 0.014 $\cup$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.07 $\cap$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.042 $\cap$
$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.02	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.002 $\cap$	$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ 0.002 $\cup$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.01 $\cap$	$\begin{bmatrix} 1 \\ 2 & 3 \end{bmatrix}$ 0.006 $\cap$

Table 22: Simplified fusion matrix version of table 21 in *sum of products* notation

$m_1 \oplus m_2$ \ m_3	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 2 \\ \end{bmatrix}$ 0.1	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.5	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 0.3
$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.56	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.056 $\cap$	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.056 $\cup$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.28 $\cap$	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.168 $\cap$
$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.28	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.028 $\cap$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.028 $\cup$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.14 $\cap$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.084 $\cap$
$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.14	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.014 $\cap$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.014 $\cup$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.07 $\cap$	$\begin{bmatrix} 1 & 3 \\ \end{bmatrix}$ 0.042 $\cap$
$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.02	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.002 $\cap$	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ 0.002 $\cup$	$\begin{bmatrix} 3 \\ \end{bmatrix}$ 0.01 $\cap$	$\begin{bmatrix} 1 \\ \end{bmatrix}$ 0.006 $\cap$

Table 23: Final result of DSmH for information from table 11

[1]	[3]	$[1 \cap 3]$	$[1 \cup 3]$
0.288	0.178	0.532	0.002

.

3.5 Performances analysis

Since no other implementation of DSmT on D^Θ is known to be written, we don't have the possibility of comparing the written code to any other code. However, we are able to track the evolution of the execution time with the growth in the number of objects and/or the number of sources. The same can be done with the memory requirement. Until another implementation of the DSmH is written, it's the only pertinent feasable performances analysis. The program usually gives us in output DSmH calculation results with plausibility and belief values. However, the tests we've realized were done on the DSmH alone. The code, which can be found in appendix, had also to be modified to output time and size of variables which can undoubdly affect time of execution and probably size required by the program.

For the measurement of the time of execution, we've only used the `tic toc` Matlab command between each tested cases. The `clear` command, which clears variables values, was also used to prevent Matlab from altering execution time.

For the size of variable measurements, we've used the `whos` command at the end of the file `hybrid.m`. The program is divided into 22 files, however the main variables are contained in `hybrid.m`. Also, most of the functions calls few other functions one into another. We can also assumes that once Matlab leaves a function, it destroys all of it's variables. We considered hence the memory size values obtained within `hybrid.m` a very good lower estimate of the required memory size.

Note also that the tests were done on a Toshiba Satellite Pro 6100 station which has a Pentium M 4 running at 1.69 GHz, 2x512 MB of RAM PC2700, and a 80 GB hard drive running at 7200 rpm.

3.5.1 Execution time vs $|\Theta|$

Figure (1) shows us evolution of the execution time versus the cardinal of Θ for $|\Theta|$ going from 3 to 9. Since there's a large number of possible testing parameters, we've choosen to perform the tests in a specific case. It consists of measuring the evolution of the execution time versus $|\Theta|$ while keeping the number of sources to 5 with the same information provided by each sources for each point. Each sources gives a bba with only six focal elements ($|\mathcal{K}| =)$. We've choosen also to put only six constraints on each points. Moreover, the constraints are dynamical and applied at each steps of combination. As we can see on figure (1), time evolves exponentially with $|\Theta|$.

3.5.2 Execution time vs Number of sources

Figure (2) shows us evolution of the execution time versus the number of sources going from 3 to 9. Since there's a large number of possible testing parameters, we've choosen to perform the tests in a specific case. It consists of measuring the evolution of the execution time versus the number of sources while keeping $|\Theta|$ to 5 with information varying for each sources for each point. Each sources gives a bba with only six focal elements ($|\mathcal{K}| =)$. We've choosen also to put only six constraints on each points, moreover, the constraints are dynamical and applied at each steps of combination. As we can see on figure (2), time also evolves exponentially with the number of sources.

3.5.3 Execution time vs $|\mathcal{K}|$

Figure (2) shows us evolution of the execution time versus the core dimension or the number of non zero mass going from 3 to 9. In this case, we've choosen to perform the tests while keeping $|\Theta|$ to 3 with a fixed number of sources of 5. We've choosen also to put only three constraints on each steps of combination. As we can see on figure (3), time also evolves almost linearly with the core dimension.

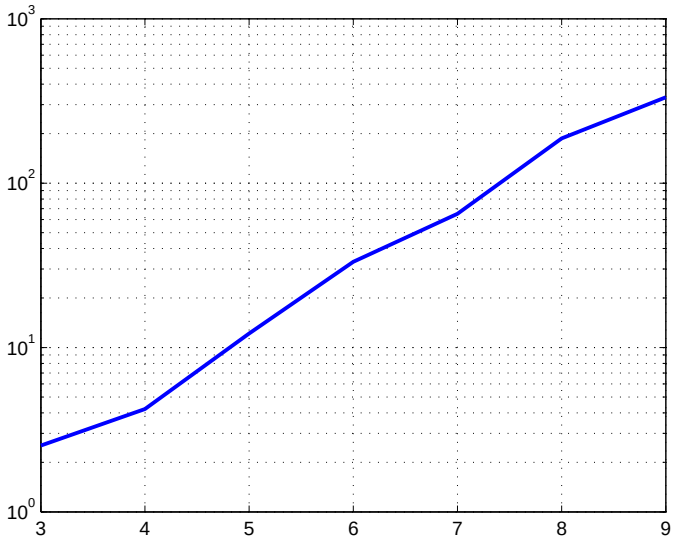


Figure 1: Evolution of execution time (sec) vs the cardinal of Θ

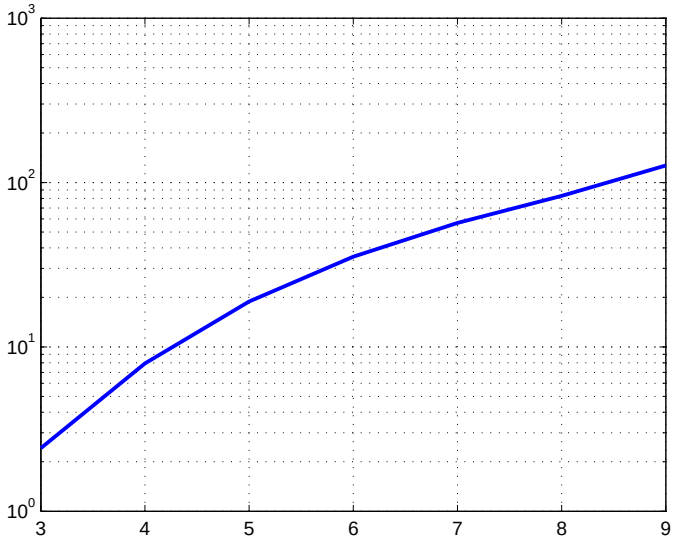


Figure 2: Evolution of execution time (sec) vs the number of sources

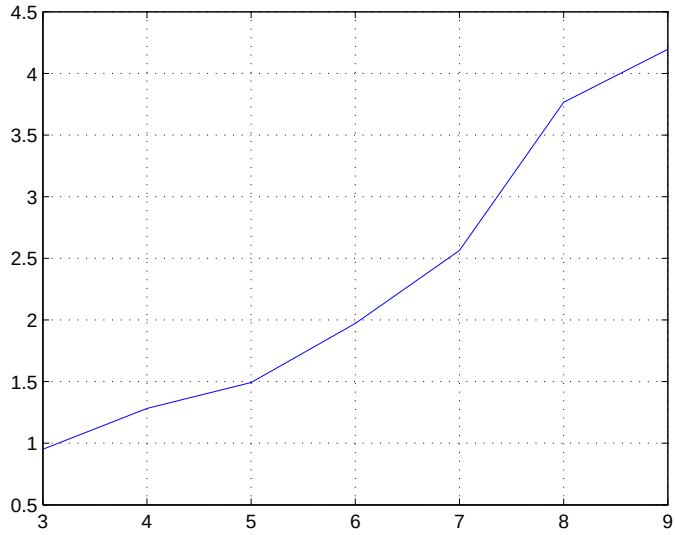


Figure 3: Evolution of execution time (sec) vs the cardinal of $\mathcal{K}$

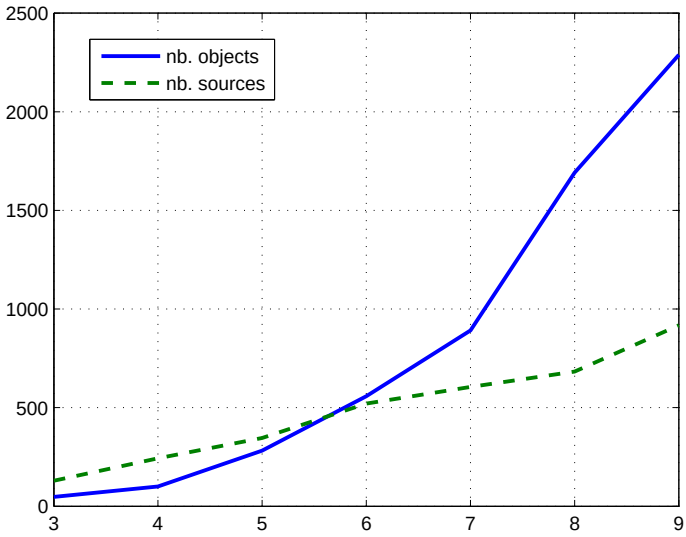


Figure 4: Evolution of memory size (KB) of hybrid.m workspace vs the number of sources or $|\Theta|$

3.5.4 Memory size vs the number of sources or $|\Theta|$

Figure (4) was realized under the same conditions as the input conditions for the execution time performances tests. We note that even with an increasing memory requirement, memory needs are still small. It is, of course, only the requirement for one of the many functions of our system. However, subdivisions of the code in many functions, the memory management system of Matlab and the fact that we only keep the necessary information to fuse helps keeping it at low levels. Also, during the tests we've observe in the Windows XP Pro task manager the amount of system memory used by Matlab. We've noted the memory use going from 79 MB before starting the test, to a peak usage a 86 MB.

We have also tryied it once in static mode with a core dimension of 10 from five sources and ten constraints with three objects in the frame of refinment to see how much memory it would take. In that simple case, we went from 79 MB (before the test started) to 137 MB (a peak memory usage during the test). A huge hunger for ressources were predictable for the static calculation mode with the enormous matrix it has to build with all the input informations.

3.5.5 Further optimization to be done

Our code's algorithm is an optimization of the original DSmH calculation process. However, certain parts of our program remains to be optimize. First of all, the possibility of rejecting information and transferring it's mass to total ignorance in the case it's mass is too small or if we have too many information should be added. Second point, at many stages of our calculation, sorting is required. As we know, sorting is one of the most time consuming process in programs and it's also the case in our program. We've used two `for` loops for sorting within two other `for` loops to go through all the elements of the matrix within the file `ordre_grandeur.m`. So as the quantity of information grows, Matlab might eventually have problems sorting the inputed information. The use of an optimized algorithm replacing this part is recommanded. There's also the possibility of using the Matlab command `sort` with some adaptations to be able to do the following sorting.

Our required sorting process in `ordre_grandeur.m` should be able to sort sets first according to the sets' size. Then, for equal sized sets, the sorting process should be able to sort in numerical order of objects. So the following set : $\theta_4 + \theta_1\theta_3\theta_4 + \theta_2\theta_3 + \theta_1\theta_3$ should be ordered this way : $\theta_4 + \theta_1\theta_3 + \theta_2\theta_3 + \theta_1\theta_3\theta_4$. A sorting process is also in use within the file `tri.m` which is able to sort matrices of sets. However the sorting process should be optimized again.

4 Conclusion

As we have seen, even with the apparent complexity of DSmH, it is still possible to engineer an efficient procedure of calculation. Such a procedure enables us to conceive an efficient Matlab code. We have conceived such a code that can perform within a reasonnable amount of time by limiting the number of `for` and `while` loops exploiting Matlab's vectorial calculation capabilities. However, even if we've obtained an optimal process of evaluating DSmH, there's still work to be done to optimize some parts of our code involving sorting.

Two avenues can be taken in the future. The first one would be to increase optimization of the actual code, trying to reduce further the number of loops, particularly in sorting. The second avenue would now be to explore how to optimize and program new combination rules such as the adaptative combination rule (ACR) [3], and the proportionnal combination rule (PCR) [3].

5 Acknowledgements

Defence R&D Canada for their financial and technical supports. Pascal Djiknavorian would also like to thank his parents and his little brother, Nejmeh, Mihran, and Edouard, for encouraging him, and for their support.

References

[1] F. Smarandache, J. Dezert (ed.), *Advances and Applications of DSmT for Information Fusion.*, American Research Press, 2004.

[2] F. Smarandache, *Unification of Fusion Theories*, ArXiv Computer Science e-prints, arXiv:cs/0409040, 2004.

[3] M.C. Florea, J. Dezert, P. Valin, F. Smarandache, A.-L. Jousselme, *Adaptative combination rule and proportional conflict redistribution rule for information fusion.* COGIS '06, Paris, 2006.

[4] M.-L. Gagnon, D. Grenier, *Rapport: Fusion de données, été 2005.*, Université Laval, 2005.

A Matlab code listing, file : aff_ensemble.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: function displaying elements and mass from a set
%
% info: elements and mass information to display
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function aff_ensemble(info)

%% for optimisation
%#inbounds
%#realonly

nI = length(info);
for k = 1 : nI

    %% displays only the non empty elements
    if ~isequal(info(k).elements,[])
        disp([num2str(info(k).elements) ' : m='  num2str(info(k).masses,'%12.8f')]);
    end
end

disp(' ')
```

B Matlab code listing, file : aff_matrice.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: function displaying elements and mass
%
% info: elements and mass information to display
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function aff_matrice(info)

%% for optimisation
%#inbounds
%#realonly

%% variables
[m,n] = size(info);

%% go through all the objects
for k = 1 : m
    for g = 1 : n
        ensemble = info(k,g).elements
        for h = 1 : length(ensemble)
            disp([num2str(ensemble{h})]);
        end
        disp ([ 'm : ' num2str(info(k,g).masses,'%6.4f') ]);
    end
end
```

C Matlab code listing, file : bon_ordre.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: function ordering vectors in sets
%
% ensembleN, ensembleM : two sets in which we have to see if some values
% are identical, if so, they must be put at the same position
%
% ensembleNOut, ensembleMOut : output vector
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ensembleMOut, ensembleNOut] = bon_ordre(ensembleM, ensembleN)

%% for optimisation
%#inbounds
%#realonly

%% variables
ensembleMOut  = {};
ensembleNOut  = {};
ensemble1     = [];
ensemble2     = [];
ensemble_temp = [];
plus_grand    = 1;

%% go through all the objects
if length(ensembleN) >= length(ensembleM)
    ensemble1  = ensembleN;
    ensemble2  = ensembleM;
    plus_grand = 1;
else
    ensemble1  = ensembleM;
    ensemble2  = ensembleN;
    plus_grand = 2;
end

%% check if there is two identical sets, otherwise check vectors
for g = 1 : length(ensemble2)
    for h = 1 : length(ensemble1)
        if isequal(ensemble1{h},ensemble2{g})
            ensemble_temp = ensemble1{g};
            ensemble1{g}  = ensemble1{h};
            ensemble1{h}  = ensemble_temp;
        end
    end
end

if isequal(plus_grand,1)
    ensembleMOut = ensemble2;
    ensembleNOut = ensemble1;
elseif isequal(plus_grand,2)
    ensembleNOut = ensemble2;
    ensembleMOut = ensemble1;
end
```

D Matlab code listing, file : calcul_DSm_hybrid_auto.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function to execute a DSm hybrid rule of combination
%               in dynamic or static mode
%
% Output: displayed in sum of product
%                                   sum for union
%                                   product for intersection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte)

%%   for optimisation
%#inbounds
%#realonly

%%   global variables
global ADD
global MULT
ADD  = -2;
MULT = -1;

%%   variables
Iall       = [];
Ihyb       = [];
contraire  = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% compute the product of sum
[contraire_complet, contraire] = faire_contraire(info);

%% case with two sources
if isequal(nombre_source,2)
    Ihyb   = hybride(info, contrainte{1},contraire,2,nombre_source,contraire_complet);
    shafer = 0;
    Iall = depart(Ihyb,2);
    Ihyb = depart(Ihyb,1);
    disp('DSm hybride');
    aff_ensemble(Ihyb);
else

    %% case with more than two sources : check the type 'sorte' of DSmH
    %% case dynamic
    if isequal(sorte,'dynamique')
       Ihyb = hybride(info,contrainte{1},contraire,2,nombre_source,contraire_complet);
        for g = 3 : nombre_source
            Ihyb          = depart(Ihyb,2);
            ensemble_step = {};
            masses_step   = [];
            disp('DSm hybride');
            aff_ensemble(Ihyb)
            for h = 1 : length(Ihyb)
```

```
                ensemble_step{h} = Ihyb(h).elements;
                masses_step(h)   = Ihyb(h).masses;
            end
            info(1).elements    = {};               info(1).masses     = [];
            info(2).elements    = {};               info(2).masses     = [];
            info(1).elements    = ensemble_step;    info(1).masses     = masses_step;
            info(2)             = info(g);
            [contraire_complet, contraire] = faire_contraire(info);
            clear Ihyb;
     Ihyb = hybride(info,contrainte{g-1},contraire,2,nombre_source,contraire_complet);
        end
        %% replace numerical value of ADD and MULT by the text 'ADD','MULT'
        Iall = depart(Ihyb,2);
        Ihyb = depart(Ihyb,1);
        disp('DSm hybride');
        aff_ensemble(Ihyb);

    %% case static
    else
        Ihyb = hybride(info,contrainte{nombre_source -1},contraire,1, ...
                                                  nombre_source,contraire_complet);
        %% replace numerical value of ADD and MULT by the text 'ADD','MULT'
        Iall = depart(Ihyb,2);
        Ihyb = depart(Ihyb,1);
        disp('DSm hybride');
        aff_ensemble(Ihyb);
    end
end

%% compute belief and plausibility
Isel = Iall;
fboe = {'pl' 'bel'};
for k=1:length(fboe)
    switch fboe{k}
        case 'pl'
            Pds = plausibilite(Isel,contrainte);
            disp('Plausibilit');
            Pds = depart(Pds,1);
            aff_ensemble(Pds);
        case 'bel'
            Bds = croyance(Isel);
            disp('Croyance');
            Bds = depart(Bds,1);
            aff_ensemble(Bds);
    end
end
```

E Matlab code listing, file : calcul_DSm_hybride.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  main file to execute a DSm hybrid rule of combination
%               in dynamic or static mode
%
% Output: displayed in sum of product
%                                   sum for union
%                                   product for intersection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
clc;

%%   for optimisation
%#inbounds
%#realonly

%%   global variables
global ADD
global MULT
ADD  = -2;
MULT = -1;

%%   variables
Iall       = [];
Ihyb       = [];
info       = [];
contrainte = [];
contraire  = [];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WRITE EVENTS AND CONSTRAINTS IN SUM OF PRODUCT NOTATION  %

 %nombre_source    = 2;
 %info(1).elements = {[1], [1, ADD, 2], [1, ADD, 3], [2], [2, ADD, 3], [3]};
 %info(1).masses   = [0.2, 0.17, 0.33, 0.03, 0.17, 0.1];
 %info(2).elements = {[1], [2], [3]};
 %info(2).masses   = [0.2, 0.4, 0.4];
 %%contrainte{1}    = {};
 %%contrainte{1}    = {[1, MULT, 2], [1, MULT, 3], [2, MULT, 3]};
 %contrainte{1}    = {[1, MULT, 2], [1, MULT, 3], [2, MULT, 3],...
 %                     [1, MULT, 2,ADD, 1, MULT, 3]...
 %                     [1, MULT, 2,ADD, 2, MULT, 3]...
 %                     [1, MULT, 3,ADD, 2, MULT, 3]...
 %                     [1, MULT, 2,ADD, 1, MULT, 3, ADD, 2, MULT, 3]};

%   nombre_source    = 4;
%   sorte            = ['dynamique'];
%%  sorte            = ['statique'];
%   info(1).elements = {[1, ADD, 2, MULT, 3], [2], [3]};
% info(1).masses   = [0.4, 0.1, 0.5];
%  info(2).elements = {[1], [2], [3]};     info(2).masses   = [0.5, 0.3, 0.2];
%  info(3).elements = {[1], [2], [3]};     info(3).masses   = [0.2, 0.4, 0.4];
```

```
%   info(4).elements = {[1], [2], [3]};     info(4).masses   = [0.4, 0.4, 0.2];
%  contrainte{1}    = {[1, MULT, 2], [1, MULT, 3], [2, MULT, 3]};
%   contrainte{2}    = {[1, MULT, 2], [1, MULT, 3], [2, MULT, 3]};
%   contrainte{3}    = {[1, MULT, 2], [1, MULT, 3], [2, MULT, 3]};

% nombre_source    = 3;   sorte    = ['dynamique'];
% info(1).elements = {[1],[3], [2, MULT, 3]};
% info(1).masses   = [0.7, 0.2, 0.1];
% info(2).elements = {[2],[1, ADD, 3], [2, ADD, 3]};
% info(2).masses   = [0.6, 0.2, 0.2];
% info(3).elements = {[1], [2], [3], [1, ADD, 2]};
% info(3).masses   = [0.1, 0.1, 0.5, 0.3];
% contrainte{1}    = {[2, MULT, 3], [2, ADD, 3]};
% contrainte{2}    = {[2], [1, MULT, 2], [2, MULT, 3],...
%                    [1, MULT, 2, ADD, 2, MULT, 3], [1, MULT, 2, MULT, 3]};

nombre_source    = 2;
info(1).elements = {[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3], [1]};
info(1).masses  =   [0.6, 0.4];
info(2).elements = {[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3], [1]};
info(2).masses  =   [0.4, 0.6];
contrainte{1}    = {};

%                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%tic;

%% compute the product of sum
[contraire_complet, contraire] = faire_contraire(info);

%% case with two sources
if isequal(nombre_source,2)
    Ihyb   = hybride(info, contrainte{1},contraire,2,nombre_source,contraire_complet);
    shafer = 0;
    Iall = depart(Ihyb,2);
    Ihyb = depart(Ihyb,1);
    disp('DSm hybride');
    aff_ensemble(Ihyb);
else

    %% case with more than two sources : check the type 'sorte' of DSmH
    %% case dynamic
    if isequal(sorte,'dynamique')
      Ihyb = hybride(info,contrainte{1},contraire,2,nombre_source,contraire_complet);
        for g = 3 : nombre_source
            Ihyb          = depart(Ihyb,2);
            ensemble_step = {};
            masses_step   = [];
            disp('DSm hybride');
            aff_ensemble(Ihyb)
            for h = 1 : length(Ihyb)
                ensemble_step{h} = Ihyb(h).elements;
                masses_step(h)   = Ihyb(h).masses;
            end
            info(1).elements    = {};               info(1).masses  = [];
            info(2).elements    = {};               info(2).masses  = [];
            info(1).elements    = ensemble_step;    info(1).masses  = masses_step;
            info(2)             = info(g);
            [contraire_complet, contraire] = faire_contraire(info);
            clear Ihyb;
            Ihyb  = hybride(info,contrainte{g-1},contraire,2,nombre_source, ...
```

```
                                                                    contraire_complet);
        end
        %% replace numerical value of ADD and MULT by the text 'ADD','MULT'
        Iall = depart(Ihyb,2);
        Ihyb = depart(Ihyb,1);
        disp('DSm hybride');
        aff_ensemble(Ihyb);

    %% case static
    else
        Ihyb = hybride(info,contrainte{nombre_source -1},contraire,1,...
                                                nombre_source,contraire_complet);
        %% replace numerical value of ADD and MULT by the text 'ADD','MULT'
        Iall = depart(Ihyb,2);
        Ihyb = depart(Ihyb,1);
        disp('DSm hybride');
        aff_ensemble(Ihyb);
    end
end

%% compute belief and plausibility
Isel = Iall;
fboe = {'pl' 'bel'};
for k=1:length(fboe)
    switch fboe{k}
        case 'pl'
            Pds = plausibilite(Isel,contrainte);
            disp('Plausibilit');
            Pds = depart(Pds,1);
            aff_ensemble(Pds);
        case 'bel'
            Bds = croyance(Isel);
            disp('Croyance');
            Bds = depart(Bds,1);
            aff_ensemble(Bds);
    end
end
%toc;
```

F Matlab code listing, file : croyance.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: function that computes belief
%
% I : final information for which we want to compute belief
%
% croyance_complet: output giving belief values and objects
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function croyance_complet = croyance(I)

%% for optimisation
%#inbounds
%#realonly

%% global variables
global ADD
global MULT
ADD  = -2;
MULT = -1;

%% variables
info           = [];
matrice_monome = [];
ignorance      = [];
nombreElement  = 0;
ensemble       = {};
vecteur1       = [];
vecteur2       = [];
f              = 1;
j              = 1;

%% separates objects, remove words ADD and MULT
for g = 1 : length(I)
    if ~isempty(I(g).elements)
        ensemble{f} = I(g).elements;
        vecteur1(f) = I(g).masses;
        vecteur2(f) = 1;
        f           = f + 1;
    end
end

info(1).elements = ensemble;
info(1).masses   = vecteur1;
info(2).elements = ensemble;
info(2).masses   = vecteur2;

[matrice_monome,ignorance,nombreElement] = separation(info,1);
matrice_monome                           = ordre_grandeur(matrice_monome,2);

%% produces the union matrix
matrice_intersection_contraire = intersection_matrice(matrice_monome,1);
matrice_intersection_contraire = ordre_grandeur(matrice_intersection_contraire,2);
```

```
matrice_intersection_contraire = dedouble(matrice_intersection_contraire,2);

%% Those for which union equals the monome (by lines), we add their masses.
[m,n] = size(matrice_intersection_contraire);
for g = 2 : m
    for h = 2 : n
        if isequal(matrice_intersection_contraire(g,h).elements,...
                                                matrice_monome(g,1).elements)
            resultat(j).elements = matrice_monome(g,1).elements;
            resultat(j).masses   = matrice_intersection_contraire(g,h).masses;
            j                    = j + 1;
        end
    end
end

croyance_complet = dedouble(resultat,1);
```

G Matlab code listing, file : dedouble.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: function that removes identical values and simplifies object
%
% matrice:  matrix to simplify, can be a set
% sorte:    indicates if input is a matrix or a set
%
% retour:   output once simplified
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [retour] = dedouble(matrice,sorte)

%% for optimisation
%#inbounds
%#realonly

%% global variable
global REPETE
REPETE = 0;

%% case set
if isequal(sorte,1)

    %% variables
    ensembleOut = [];
    j           = 1;

    %% go through elements of the set
    for g = 1 : length(matrice)
        for h = g + 1 : length(matrice)
            if isequal(matrice(h).elements,matrice(g).elements)
                matrice(h).elements = REPETE;
                matrice(g).masses   = matrice(g).masses + matrice(h).masses;
            end
        end

        if ~isequal(matrice(g).elements,REPETE) & ~isequal(matrice(g).masses,0)
            ensembleOut(j).elements = matrice(g).elements;
            ensembleOut(j).masses   = matrice(g).masses;
            j                       = j + 1;
        end
    end

    retour = ensembleOut;

%% case matrix
else

    %% variables
    [m,n]    = size(matrice);
    vecteur1 = [];
    vecteur2 = [];
```

```
if m > 1
    u = 2;
    y = 2;
else
    u = 1;
    y = 1;
end

%% go through elements of the matrix
for h = u : m
    for g = y : n
        ensemble        = {};
        ensemble        = matrice(h,g).elements;
        j               = 1;
        nouvel_ensemble = {};

        %% go through all vectors of the matrix
        for k = 1 : length(ensemble)
            vecteur1 = ensemble{k};
            if ~isempty(vecteur1)
                for f = k + 1 : length(ensemble)
                    vecteur2 = ensemble{f};

                    %% check if there is two identical vectors
                    if ~isempty(vecteur2)
                        if isequal(vecteur1, vecteur2)
                            vecteur1 = REPETE;
                        else

                            %% check if a vector is included in another
                            %% 2 intersection 2union3  : remove 2union3
                            compris = 0;
                            for v = 1 : length(vecteur1)
                                for c = 1 : length(vecteur2)
                                    if isequal(vecteur1(v),vecteur2(c))
                                        compris = compris + 1;
                                    end
                                end
                            end

                            if length(vecteur1) < length(vecteur2)
                                if isequal(compris, length(vecteur1))
                                    vecteur2 = REPETE;
                                end
                            else
                                if isequal(compris, length(vecteur2))
                                    vecteur1 = REPETE;
                                end
                            end
                        end
                    ensemble{f} = vecteur2;
                    end
                end
                ensemble{k} = vecteur1;
            end

            if ~isequal(ensemble{k},REPETE)
                nouvel_ensemble{j} = ensemble{k};
                j                  = j + 1;
            end
        end
```

```
            matriceOut(h,g).elements = nouvel_ensemble;
            matriceOut(h,g).masses   = matrice(h,g).masses;
        end
    end
    matriceOut = tri(matriceOut,1);

    retour = matriceOut;
end
```

H Matlab code listing, file : depart.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function putting ADD and MULT
%
% ensemble_complet: set for which we want to add ADD and MULT
%       each element is a cell including vectors
%       each vector is a product and a change of vector is a sum
% sorte: to know if it has to be in numerical value or not
%
% ensemble_final: output with ADD and MULT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ensemble_final] = depart(ensemble_complet,sorte)

%% for optimisation
%#inbounds
%#realonly

%% global variables
global A
global M
global ADDS
global MULTS
ADDS  = ' ADD ';
MULTS = ' MULT ';
A     = -2;
M     = -1;

%% variables
ensemble       = [];
ensemble_final = [];

%% go through vectors of the set
for g = 1 : length(ensemble_complet)
        ensemble = ensemble_complet(g).elements;
        for k = 1 : length(ensemble)

            %% first time
            if isequal(k,1)
                if isequal(length(ensemble{k}),1)
                    if isequal(sorte,1)
                        ensemble_final(g).elements = [num2str(ensemble{1})];
                    else
                        ensemble_final(g).elements = [ensemble{1}];
                    end
                else
                    vecteur = ensemble{k};
                    for f = 1 : length(vecteur)
                        if isequal(f,1)
                            if isequal(sorte,1)
                                ensemble_final(g).elements = [num2str(vecteur(f))];
                            else
                                ensemble_final(g).elements = [vecteur(f)];
```

```
                end
            else
        if isequal(sorte,1)
            ensemble_final(g).elements = [...
                                    ensemble_final(g).elements, ...
                                    MULTS, num2str(vecteur(f))];
        else
            ensemble_final(g).elements = [...
                                    ensemble_final(g).elements, ...
                                    M, vecteur(f)];
        end
            end
        end
    end

%% puts ' ADD ' since change of vector
else
    if isequal(sorte,1)
        ensemble_final(g).elements = ...
         [ensemble_final(g).elements, ADDS];
    else
        ensemble_final(g).elements = ...
         [ensemble_final(g).elements, A];
    end

    if isequal(length(ensemble{k}),1)
        if isequal(sorte,1)
            ensemble_final(g).elements = ...
             [ensemble_final(g).elements, ...
             num2str(ensemble{k})];
        else
            ensemble_final(g).elements = ...
             [ensemble_final(g).elements, ...
             ensemble{k}];
        end

    %% puts ' MULT '
    else
        premier = 1;
        vecteur = ensemble{k};
        for f = 1 : length(vecteur)
                if premier == 1
                    if isequal(sorte,1)
                        ensemble_final(g).elements = ...
                         [ensemble_final(g).elements,...
                          num2str(vecteur(f))];
                    else
                        ensemble_final(g).elements = ...
                         [ensemble_final(g).elements, ...
                         vecteur(f)];
                    end
                    premier                   = 0;
                else
                    if isequal(sorte,1)
                        ensemble_final(g).elements = ...
                         [ensemble_final(g).elements, ...
                         MULTS, num2str(vecteur(f))];
                    else
                        ensemble_final(g).elements = ...
                         [ensemble_final(g).elements, ...
                         M, vecteur(f)];
```

```
                                end
                            end
                        end
                    end
                end
            end
        ensemble_final(g).masses = ensemble_complet(g).masses;
    end
```

I Matlab code listing, file : DSmH_auto.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% description: file from which we can call the function version of the DSmH
%
%%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
% The examples in this file were taken from:
%''Advances and Applications of DSmT for Information Fusion''
% written by par Jean Dezert and Florentin Smarandache, 2004
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
clc;
info       = [];
contrainte = [];

global ADD
global MULT
ADD  = -2;
MULT = -1;

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 1, Page 21');

 nombre_source    = 2;
 sorte            = ['dynamique'];
 info(1).elements = {[1],[2],[3],[1, ADD, 2]};
 info(1).masses   = [0.1, 0.4, 0.2, 0.3];
 info(2).elements = {[1],[2],[3],[1, ADD, 2]};
 info(2).masses   = [0.5, 0.1, 0.3, 0.1];
 contrainte{1}    = {[1, MULT, 2, MULT, 3],[1, MULT, 2],[2, MULT, 3],...
  [1, MULT, 3],[3],[1, MULT, 3, ADD, 2, MULT, 3],...
   [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 2, Page 25');

 nombre_source    = 2;
 info(1).elements = {[1],[3],[4]};
```

```
 info(1).masses   = [0.998, 0.001, 0.001];
 info(2).elements =  {[2],[4]};
 info(2).masses   = [0.998, 0.002];
 contrainte{1}    = {[1, MULT, 3],[1, MULT, 2],[1, MULT, 4],[2, MULT, 3],...
 [2, MULT, 4],[3, MULT, 4],[1, MULT, 2, MULT, 3],[1, MULT, 2, MULT, 4],...
 [1, MULT, 3, MULT, 4],[2, MULT, 3, MULT, 4],[1, MULT, 2, MULT, 3, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 4],[2, MULT, 3, ADD, 2, MULT, 4],...
 [1, MULT, 2, ADD, 2, MULT, 4],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 3, ADD, 3, MULT, 4],...
 [2, MULT, 3, ADD, 3, MULT, 4],[1, MULT, 4, ADD, 2, MULT, 4],...
 [1, MULT, 4, ADD, 3, MULT, 4],[2, MULT, 4, ADD, 3, MULT, 4],...
 [1, MULT, 2, MULT, 3, ADD, 1, MULT, 2, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4, ADD, 2, MULT, 3, ADD, 2, MULT, 4]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 4, Page 27');

 nombre_source    = 2;
 info(1).elements = {[1],[3, ADD, 4]};
 info(1).masses   = [0.99, 0.01];
 info(2).elements =  {[2],[3, ADD, 4]};
 info(2).masses   = [0.98, 0.02];
 contrainte{1}    = {[1, MULT, 3],[1, MULT, 2],[1, MULT, 4],[2, MULT, 3],...
 [2, MULT, 4],[3, MULT, 4],[1, MULT, 2, MULT, 3],[1, MULT, 2, MULT, 4],...
 [1, MULT, 3, MULT, 4],[2, MULT, 3, MULT, 4],[1, MULT, 2, MULT, 3, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 4],[2, MULT, 3, ADD, 2, MULT, 4],...
 [1, MULT, 2, ADD, 2, MULT, 4],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 3, ADD, 3, MULT, 4],...
 [2, MULT, 3, ADD, 3, MULT, 4],[1, MULT, 4, ADD, 2, MULT, 4],...
 [1, MULT, 4, ADD, 3, MULT, 4],[2, MULT, 4, ADD, 3, MULT, 4],...
 [1, MULT, 2, MULT, 3, ADD, 1, MULT, 2, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4, ADD, 2, MULT, 3, ADD, 2, MULT, 4]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5, Page 28');

 nombre_source    = 2;
 info(1).elements = {[1],[2],[1, MULT, 2]};
 info(1).masses   = [0.5, 0.1, 0.4];
 info(2).elements =  {[1],[2],[1, MULT, 2]};
 info(2).masses   = [0.1, 0.6, 0.3];
 contrainte{1}    = {[1, MULT, 2]};
```

```
    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 1, Page 80');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 3],[3],[1, MULT, 2],[2],[1],...
                                        [1, ADD, 3],[1, ADD, 2]};
 info(1).masses   = [0.1, 0.3, 0.1, 0.2, 0.1, 0.1, 0.1];
 info(2).elements = {[2, MULT, 3],[3],[1, MULT, 2],[2],[1],[1, ADD, 3]};
 info(2).masses   = [0.2, 0.1, 0.2, 0.1, 0.2, 0.2];
 contrainte{1}    = {[1, MULT, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 2, Page 81');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 3],[3],[1, MULT, 2],[2],[1],...
                                      [1, ADD, 3],[1, ADD, 2]};
 info(1).masses   = [0.1, 0.3, 0.1, 0.2, 0.1, 0.1, 0.1];
 info(2).elements = {[2, MULT, 3],[3],[1, MULT, 2],[2],[1],[1, ADD, 3]};
 info(2).masses   = [0.2, 0.1, 0.2, 0.1, 0.2, 0.2];
 contrainte{1}    = {[1, MULT, 2], [1, MULT, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 3, Page 83');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 3],[3],[1, MULT, 2],[2],[1],...
                                          [1, ADD, 3],[1, ADD, 2]};
 info(1).masses   = [0.1, 0.3, 0.1, 0.2, 0.1, 0.1, 0.1];
 info(2).elements = {[2, MULT, 3],[3],[1, MULT, 2],[2],[1],[1, ADD, 3]};
 info(2).masses   = [0.2, 0.1, 0.2, 0.1, 0.2, 0.2];
 contrainte{1}    = {[2, MULT, 3], [1, MULT, 2, MULT, 3],...
                       [1, MULT, 2, ADD, 2, MULT, 3], [1, MULT, 2]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 4, Page 85');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 3],[3],[1, MULT, 2],[2],[1],...
                     [1, ADD, 3],[1, ADD, 2]};
 info(1).masses   = [0.1, 0.3, 0.1, 0.2, 0.1, 0.1, 0.1];
 info(2).elements = {[2, MULT, 3],[3],[1, MULT, 2],[2],[1],[1, ADD, 3]};
 info(2).masses   = [0.2, 0.1, 0.2, 0.1, 0.2, 0.2];
 contrainte{1}    = {[1, MULT, 3], [1, MULT, 2, MULT, 3],[2, MULT, 3],...
                     [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 2],...
                     [1, MULT, 2, ADD, 2, MULT, 3],...
                     [1, MULT, 2, ADD, 1, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5, Page 86');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 3],[3],[1, MULT, 2],[2],[1],...
                     [1, ADD, 3],[1, ADD, 2]};
 info(1).masses   = [0.1, 0.3, 0.1, 0.2, 0.1, 0.1, 0.1];
 info(2).elements = {[2, MULT, 3],[3],[1, MULT, 2],[2],[1],[1, ADD, 3]};
 info(2).masses   = [0.2, 0.1, 0.2, 0.1, 0.2, 0.2];
 contrainte{1}    = {[1, MULT, 3], [1, MULT, 2, MULT, 3],[1],...
                     [1, MULT, 2],[1, MULT, 2, ADD, 1, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 6, Page 87');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 3],[3],[1, MULT, 2],[2],[1],...
                     [1, ADD, 3],[1, ADD, 2]};
 info(1).masses   = [0.1, 0.3, 0.1, 0.2, 0.1, 0.1, 0.1];
 info(2).elements = {[2, MULT, 3],[3],[1, MULT, 2],[2],[1],[1, ADD, 3]};
 info(2).masses   = [0.2, 0.1, 0.2, 0.1, 0.2, 0.2];
 contrainte{1}    = {[1, MULT, 2, MULT, 3], [2, MULT, 3], [1, MULT, 3], ...
  [1, MULT, 3, ADD, 2, MULT, 3], [1, MULT, 2], [1, MULT, 2, ADD, 2, MULT, 3],...
  [1, MULT, 2, ADD, 1, MULT, 3], [1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
  [2], [2, ADD, 1, MULT, 3], [1], [1, ADD, 2, MULT, 3],[1, ADD, 2]};
```

```
    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 7, Page 88');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 3],[3],[1, MULT, 2],[2],[1],...
                     [1, ADD, 3],[1, ADD, 2]};
 info(1).masses   = [0.1, 0.3, 0.1, 0.2, 0.1, 0.1, 0.1];
 info(2).elements = {[2, MULT, 3],[3],[1, MULT, 2],[2],[1],[1, ADD, 3]};
 info(2).masses   = [0.2, 0.1, 0.2, 0.1, 0.2, 0.2];
 contrainte{1}    = {[1, MULT, 2, MULT, 3], [2, MULT, 3], [1, MULT, 3],...
  [1, MULT, 3, ADD, 2, MULT, 3], [3], [1, MULT, 2], ...
  [1, MULT, 2, ADD, 2, MULT, 3], [1, MULT, 2, ADD, 1, MULT, 3],...
   [1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3], [3, ADD, 1, MULT, 2]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 1, Page 90');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(1).masses   = [0.01,0.04,0.03,0.01,0.03,0.02,0.02,0.03,0.04,0.04,...
                     0.02,0.01,0.2,0.01,0.02,0.04,0.03,0.4];
 info(2).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(2).masses   = [0.4,0.03,0.04,0.02,0.04,0.20,0.01,0.04,0.03,...
                     0.03,0.01,0.02,0.02,0.02,0.01,0.03,0.04,0.01];
 contrainte{1}    = {[1, MULT, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 2, Page 90');
```

```
 nombre_source    = 2;
 info(1).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(1).masses   = [0.01,0.04,0.03,0.01,0.03,0.02,0.02,0.03,0.04,...
 0.04,0.02,0.01,0.2,0.01,0.02,0.04,0.03,0.4];
 info(2).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(2).masses   = [0.4,0.03,0.04,0.02,0.04,0.20,0.01,0.04,0.03,0.03,...
 0.01,0.02,0.02,0.02,0.01,0.03,0.04,0.01];
 contrainte{1}    = {[1, MULT, 2], [1, MULT, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 3, Page 90');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(1).masses   = [0.01,0.04,0.03,0.01,0.03,0.02,0.02,0.03,0.04,0.04,...
 0.02,0.01,0.2,0.01,0.02,0.04,0.03,0.4];
 info(2).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(2).masses   = [0.4,0.03,0.04,0.02,0.04,0.20,0.01,0.04,0.03,0.03,0.01,...
 0.02,0.02,0.02,0.01,0.03,0.04,0.01];
 contrainte{1}    = {[1, MULT, 2], [1, MULT, 2, MULT, 3],[2, MULT, 3],...
 [1, MULT, 2, ADD, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 4, Page 90');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
```

```
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(1).masses   = [0.01,0.04,0.03,0.01,0.03,0.02,0.02,0.03,0.04,0.04,0.02,...
 0.01,0.2,0.01,0.02,0.04,0.03,0.4];
 info(2).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(2).masses   = [0.4,0.03,0.04,0.02,0.04,0.20,0.01,0.04,0.03,...
 0.03,0.01,0.02,0.02,0.02,0.01,0.03,0.04,0.01];
 contrainte{1}    = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5, Page 90');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(1).masses   = [0.01,0.04,0.03,0.01,0.03,0.02,0.02,0.03,0.04,...
 0.04,0.02,0.01,0.2,0.01,0.02,0.04,0.03,0.4];
 info(2).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(2).masses   = [0.4,0.03,0.04,0.02,0.04,0.20,0.01,0.04,0.03,0.03,...
 0.01,0.02,0.02,0.02,0.01,0.03,0.04,0.01];
 contrainte{1}    = {[1, MULT, 3], [1, MULT, 2, MULT, 3],[1],...
                     [1, MULT, 2],[1, MULT, 2, ADD, 1, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 6, Page 90');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
```

```
 info(1).masses   = [0.01,0.04,0.03,0.01,0.03,0.02,0.02,0.03,0.04,0.04,...
 0.02,0.01,0.2,0.01,0.02,0.04,0.03,0.4];
 info(2).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(2).masses   = [0.4,0.03,0.04,0.02,0.04,0.20,0.01,0.04,0.03,0.03,...
 0.01,0.02,0.02,0.02,0.01,0.03,0.04,0.01];
 contrainte{1}    = {[1, MULT, 2, MULT, 3], [2, MULT, 3], [1, MULT, 3], ...
 [1, MULT, 3, ADD, 2, MULT, 3], [1, MULT, 2], [1, MULT, 2, ADD, 2, MULT, 3],...
  [1, MULT, 2, ADD, 1, MULT, 3], [1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
   [2], [2, ADD, 1, MULT, 3], [1], [1, ADD, 2, MULT, 3],[1, ADD, 2]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 7, Page 90');

 nombre_source    = 2;
 info(1).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
 [3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1],...
 [1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(1).masses   = [0.01,0.04,0.03,0.01,0.03,0.02,0.02,0.03,0.04,...
 0.04,0.02,0.01,0.2,0.01,0.02,0.04,0.03,0.4];
 info(2).elements = {[1, MULT, 2, MULT, 3],[2, MULT, 3],[1, MULT, 3],...
[1, MULT, 3, ADD, 2, MULT, 3],[3],[1, MULT, 2],[1, MULT, 2, ADD, 2, MULT, 3],...
[1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3],...
[3, ADD, 1, MULT, 2],[2],[2, ADD, 1, MULT, 3],[2, ADD, 3],[1], ...
[1, ADD, 2, MULT, 3],[1, ADD, 3],[1, ADD, 2],[1, ADD, 2, ADD, 3]};
 info(2).masses   = [0.4,0.03,0.04,0.02,0.04,0.20,0.01,0.04,0.03,0.03,...
 0.01,0.02,0.02,0.02,0.01,0.03,0.04,0.01];
 contrainte{1}    = {[1, MULT, 2, MULT, 3], [2, MULT, 3], [1, MULT, 3], ...
 [1, MULT, 3, ADD, 2, MULT, 3], [3], [1, MULT, 2], ...
 [1, MULT, 2, ADD, 2, MULT, 3],  [1, MULT, 2, ADD, 1, MULT, 3], ...
 [1, MULT, 2, ADD, 1, MULT, 3, ADD, 2, MULT, 3], [3, ADD, 1, MULT, 2]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 1, Pages 93-94');

 nombre_source    = 2;
 info(1).elements = {[1], [3]};
 info(1).masses   = [0.5, 0.5];
 info(2).elements = {[2], [3]};
 info(2).masses   = [0.5, 0.5];
```

```
 contrainte{1}    = {[1, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 3.2, Page 97');

 nombre_source    = 3;
 %sorte            = ['dynamique'];
 sorte            = ['statique'];
 info(1).elements = {[1],[2],[1, ADD, 2],[1, MULT, 2]};
 info(1).masses   = [0.1, 0.2, 0.3, 0.4];
 info(2).elements = {[1],[2],[1, ADD, 2],[1, MULT, 2]};
 info(2).masses   = [0.5, 0.3, 0.1, 0.1];
 info(3).elements = {[3],[1, MULT, 3],[2, ADD, 3]};
 info(3).masses   = [0.4, 0.3, 0.3];
 contrainte{1}    = {};
 contrainte{2}    = {[3],[1, MULT, 2, MULT, 3],[1, MULT, 3],...
 [2, MULT, 3],[1, MULT, 3, ADD, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 3.4, Page 99');

 nombre_source    = 3;
 sorte            = ['dynamique'];
 %sorte            = ['statique'];
 info(1).elements = {[1],[2]};
 info(1).masses   = [0.6, 0.4];
 info(2).elements = {[1],[2]};
 info(2).masses   = [0.7, 0.3];
 info(3).elements = {[1], [2], [3]};
 info(3).masses   = [0.5, 0.2, 0.3];
 contrainte{1}    = {};
 contrainte{2}    = {[1, MULT, 2, MULT, 3],[1, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 3.5, Pages 99-100');

 nombre_source    = 3;
```

```
 sorte            = ['dynamique'];
 %sorte            = ['statique'];
 info(1).elements = {[1],[2]};
 info(1).masses   = [0.6, 0.4];
 info(2).elements = {[1],[2]};
 info(2).masses   = [0.7, 0.3];
 info(3).elements = {[1], [2], [3]};
 info(3).masses   = [0.5, 0.2, 0.3];
 contrainte{1}    = {};
 contrainte{2}    = {[3], [1, MULT, 3], [2, MULT, 3], ...
 [1, MULT, 2, MULT, 3], [1, MULT, 3, ADD, 2, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 3.6, Page 100');

 nombre_source    = 2;
 info(1).elements = {[1], [2], [1, MULT, 2]};
 info(1).masses   = [0.5, 0.4 0.1];
 info(2).elements = {[1], [2], [1, MULT, 3], [4]};
 info(2).masses   = [0.3, 0.2, 0.1, 0.4];
 contrainte{1}    = {[1, MULT, 3], [1, MULT, 2], [1, MULT, 3, MULT, 4],...
  [1, MULT, 2, MULT, 3], [1, MULT, 2, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 3.7, Page 101');

 nombre_source    = 2;
 info(1).elements = {[1], [2], [1, MULT, 2], [1, MULT, 3], [4]};
 info(1).masses   = [0.2, 0.4, 0.1, 0.2, 0.1];
 info(2).elements = {[1], [2], [1, MULT, 2], [1, MULT, 3], [4]};
 info(2).masses   = [0.1, 0.3, 0.2, 0.1, 0.3];
 contrainte{1}    = {[1, MULT, 3], [1, MULT, 2], [1, MULT, 3, MULT, 4],...
  [1, MULT, 2, MULT, 3], [1, MULT, 2, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5.2.1.2, Page 106');
```

```
 nombre_source    = 2;
 info(1).elements = {[1], [3]};
 info(1).masses   = [0.6, 0.4];
 info(2).elements = {[2], [4]};
 info(2).masses   = [0.2, 0.8];
 contrainte{1}    = {[1, MULT, 3],[1, MULT, 2],[1, MULT, 4],[2, MULT, 3],...
 [2, MULT, 4],[3, MULT, 4],[1, MULT, 2, MULT, 3],[1, MULT, 2, MULT, 4],...
 [1, MULT, 3, MULT, 4],[2, MULT, 3, MULT, 4],[1, MULT, 2, MULT, 3, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 4],[2, MULT, 3, ADD, 2, MULT, 4],...
 [1, MULT, 2, ADD, 2, MULT, 4],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 3, ADD, 3, MULT, 4],...
 [2, MULT, 3, ADD, 3, MULT, 4],[1, MULT, 4, ADD, 2, MULT, 4],...
 [1, MULT, 4, ADD, 3, MULT, 4],[2, MULT, 4, ADD, 3, MULT, 4],...
 [1, MULT, 2, MULT, 3, ADD, 1, MULT, 2, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4, ADD, 2, MULT, 3, ADD, 2, MULT, 4]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5.2.1.3, Page 107');

 nombre_source    = 3;
 sorte            = ['dynamique'];
 %rep du livre = statique
 sorte            = ['statique'];
 info(1).elements = {[1],[3]};
 info(1).masses   = [0.6, 0.4];
 info(2).elements = {[2],[4]};
 info(2).masses   = [0.2, 0.8];
 info(3).elements = {[2], [4]};
 info(3).masses   = [0.3, 0.7];
 contrainte{1}    = {};
 contrainte{2}   = {[1, MULT, 3],[1, MULT, 2],[1, MULT, 4],[2, MULT, 3],...
 [2, MULT, 4],[3, MULT, 4],[1, MULT, 2, MULT, 3],[1, MULT, 2, MULT, 4],...
 [1, MULT, 3, MULT, 4],[2, MULT, 3, MULT, 4],[1, MULT, 2, MULT, 3, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 4],[2, MULT, 3, ADD, 2, MULT, 4],...
 [1, MULT, 2, ADD, 2, MULT, 4],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 3, ADD, 3, MULT, 4],...
 [2, MULT, 3, ADD, 3, MULT, 4],[1, MULT, 4, ADD, 2, MULT, 4],...
 [1, MULT, 4, ADD, 3, MULT, 4],[2, MULT, 4, ADD, 3, MULT, 4],...
 [1, MULT, 2, MULT, 3, ADD, 1, MULT, 2, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4, ADD, 2, MULT, 3, ADD, 2, MULT, 4]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5.2.2.1, Page 108');
```

```
 nombre_source    = 2;
 info(1).elements = {[1], [2], [1, ADD, 2]};
 info(1).masses   = [0.4, 0.5, 0.1];
 info(2).elements = {[3], [4]};
 info(2).masses   = [0.3, 0.7];
 contrainte{1}    = {[1, MULT, 3],[1, MULT, 2],[1, MULT, 4],[2, MULT, 3],...
 [2, MULT, 4],[3, MULT, 4],[1, MULT, 2, MULT, 3],[1, MULT, 2, MULT, 4],...
 [1, MULT, 3, MULT, 4],[2, MULT, 3, MULT, 4],[1, MULT, 2, MULT, 3, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 4],[2, MULT, 3, ADD, 2, MULT, 4],...
 [1, MULT, 2, ADD, 2, MULT, 4],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 3, ADD, 3, MULT, 4],...
 [2, MULT, 3, ADD, 3, MULT, 4],[1, MULT, 4, ADD, 2, MULT, 4],...
 [1, MULT, 4, ADD, 3, MULT, 4],[2, MULT, 4, ADD, 3, MULT, 4],...
 [1, MULT, 2, MULT, 3, ADD, 1, MULT, 2, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4, ADD, 2, MULT, 3, ADD, 2, MULT, 4]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5.2.2.2, Page 109');

 nombre_source    = 3;
 sorte           = ['dynamique'];
 %sorte            = ['statique'];
 info(3).elements = {[1],[2],[1, ADD, 2]};
 info(3).masses   = [0.4, 0.5, 0.1];
 info(2).elements = {[3],[4],[3, ADD, 4]};
 info(2).masses   = [0.3, 0.6, 0.1];
 info(1).elements = {[1], [1, ADD, 2]};
 info(1).masses   = [0.8, 0.2];
 contrainte{1}    = {};
 contrainte{2}    = {[1, MULT, 3],[1, MULT, 2],[1, MULT, 4],...
 [2, MULT, 3],[2, MULT, 4],[3, MULT, 4],[1, MULT, 2, MULT, 3],...
 [1, MULT, 2, MULT, 4],[1, MULT, 3, MULT, 4],[2, MULT, 3, MULT, 4],...
 [1, MULT, 2, MULT, 3, MULT, 4],[1, MULT, 3, ADD, 1, MULT, 4],...
 [1, MULT, 2, ADD, 1, MULT, 3],[1, MULT, 2, ADD, 1, MULT, 4],...
 [2, MULT, 3, ADD, 2, MULT, 4],[1, MULT, 2, ADD, 2, MULT, 4],...
 [1, MULT, 2, ADD, 2, MULT, 3],[1, MULT, 3, ADD, 2, MULT, 3],...
 [1, MULT, 3, ADD, 3, MULT, 4],[2, MULT, 3, ADD, 3, MULT, 4],...
 [1, MULT, 4, ADD, 2, MULT, 4],[1, MULT, 4, ADD, 3, MULT, 4],...
 [2, MULT, 4, ADD, 3, MULT, 4],[1, MULT, 2, MULT, 3, ADD, 1, MULT, 2, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4, ADD, 2, MULT, 3, ADD, 2, MULT, 4]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5.4.1, Page 116');
```

```
 nombre_source    = 2;
 info(1).elements = {[1], [3, ADD, 4]};
 info(1).masses   = [0.99, 0.01];
 info(2).elements = {[2], [3, ADD, 4]};
 info(2).masses   = [0.98, 0.02];
 contrainte{1}    = {[1, MULT, 3],[1, MULT, 2],[1, MULT, 4],[2, MULT, 3],...
 [2, MULT, 4],[3, MULT, 4],[1, MULT, 2, MULT, 3],[1, MULT, 2, MULT, 4],...
 [1, MULT, 3, MULT, 4],[2, MULT, 3, MULT, 4],[1, MULT, 2, MULT, 3, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 4],[2, MULT, 3, ADD, 2, MULT, 4],...
 [1, MULT, 2, ADD, 2, MULT, 4],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 3, ADD, 3, MULT, 4],...
 [2, MULT, 3, ADD, 3, MULT, 4],[1, MULT, 4, ADD, 2, MULT, 4],...
 [1, MULT, 4, ADD, 3, MULT, 4],[2, MULT, 4, ADD, 3, MULT, 4],...
 [1, MULT, 2, MULT, 3, ADD, 1, MULT, 2, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4, ADD, 2, MULT, 3, ADD, 2, MULT, 4]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5.4.2, Page 116');

 nombre_source    = 3;
 %sorte            = ['dynamique'];
 sorte            = ['statique'];
 info(1).elements = {[1],[4, ADD, 5]};
 info(1).masses   = [0.99, 0.01];
 info(3).elements = {[2],[3],[4, ADD, 5]};
 info(3).masses   = [0.98, 0.01, 0.01];
 info(2).elements = {[1], [2], [3], [4, ADD, 5]};
 info(2).masses   = [0.01, 0.01, 0.97, 0.01];
 contrainte{1}    = {};
 contrainte{2}    = {};
 contrainte{2}    = {[1, MULT, 3],[1, MULT, 2],[1, MULT, 4],[2, MULT, 3],...
 [2, MULT, 4],[3, MULT, 4],[1, MULT, 2, MULT, 3],[1, MULT, 2, MULT, 4],...
 [1, MULT, 3, MULT, 4],[2, MULT, 3, MULT, 4],[1, MULT, 2, MULT, 3, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4],[1, MULT, 2, ADD, 1, MULT, 3],...
 [1, MULT, 2, ADD, 1, MULT, 4],[2, MULT, 3, ADD, 2, MULT, 4],...
 [1, MULT, 2, ADD, 2, MULT, 4],[1, MULT, 2, ADD, 2, MULT, 3],...
 [1, MULT, 3, ADD, 2, MULT, 3],[1, MULT, 3, ADD, 3, MULT, 4],...
 [2, MULT, 3, ADD, 3, MULT, 4],[1, MULT, 4, ADD, 2, MULT, 4],...
 [1, MULT, 4, ADD, 3, MULT, 4],[2, MULT, 4, ADD, 3, MULT, 4],...
 [1, MULT, 2, MULT, 3, ADD, 1, MULT, 2, MULT, 4],...
 [1, MULT, 3, ADD, 1, MULT, 4, ADD, 2, MULT, 3, ADD, 2, MULT, 4],...
  [1, MULT, 5],[2, MULT, 5],[3, MULT, 5],[4, MULT, 5],[1, MULT, 2, MULT, 5],...
  [1, MULT, 3, MULT, 5],[1, MULT, 4, MULT, 5],[2, MULT, 3, MULT, 5],...
  [2, MULT, 4, MULT, 5],[3, MULT, 4, MULT, 5],[1, MULT, 2, MULT, 3, MULT, 5],...
  [1, MULT, 2, MULT, 4, MULT, 5],[1, MULT, 3, MULT, 4, MULT, 5],...
  [2, MULT, 3, MULT, 4, MULT, 5],[1, MULT, 2, MULT, 3, MULT, 4, MULT, 5],...
  [1, MULT, 4, ADD, 1, MULT, 5],[2, MULT, 4, ADD, 2, MULT, 5],...
  [3, MULT, 4, ADD, 3, MULT, 5],[1, MULT, 2, MULT, 4, ADD, 1, MULT, 2, MULT, 5],...
  [1, MULT, 3, MULT, 4, ADD, 1, MULT, 3, MULT, 5],...
  [2, MULT, 3, MULT, 4, ADD, 2, MULT, 3, MULT, 5],...
  [1, MULT, 2, MULT, 3, MULT, 4, ADD, 1, MULT, 2, MULT, 3, MULT, 5]};
```

```
    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
    info       = [];
    contrainte = [];
    disp('Example 5.5.1, Page 118');

 nombre_source    = 2;
 info(1).elements = {[1], [3], [4, ADD, 5], [5, ADD, 6]};
 info(1).masses   = [0.3, 0.4, 0.2, 0.1];
 info(2).elements = {[2]};
 info(2).masses   = [1];
 contrainte{1}    = {[1, MULT, 2],[2, MULT, 3],...
 [2, MULT, 4, ADD, 2, MULT, 5],[2, MULT, 5, ADD, 2, MULT, 6]};


    calcul_DSm_hybrid_auto(nombre_source, sorte, info, contrainte);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    disp(' ');
```

J Matlab code listing, file : enlever_contrainte.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% description: function removing constraints in sets
%
% ensemble_complet:  sets composed of S1, S2, S3
% contrainte_separe: constraints' sets : divided in cells with vectors :
%                    each vector is a product, and a change of vector = sum
%
% ensemble_complet: final set
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ensemble_complet] = ...
                enlever_contrainte(ensemble_complet, contrainte_separe);

%pour optimiser
%#inbounds
%#realonly

%variables globales
global ENLEVE
ENLEVE = {};

%variables
ensemble_contrainte = {};
ensemble_elements   = [];
ensemble_produit    = [];

%parcourir toutes les contraintes
for g = 1 : length(contrainte_separe)
    ensemble_contrainte = contrainte_separe{g};
    for h = 1 : length(ensemble_complet)
        %si la contrainte est en entier dans l'ensemble complet, l'enlever
        if isequal(ensemble_contrainte, ensemble_complet(h).elements)
            ensemble_complet(h).elements = ENLEVE;
            ensemble_complet(h).masses   = 0;
        end
    end
end

%parcourir les contraintes
for g = 1 : length(contrainte_separe)
    ensemble_contrainte = contrainte_separe{g};
    %si elle est un singleton
    if isequal(length(ensemble_contrainte),1) & ...
                                isequal(length(ensemble_contrainte{1}),1)

        for h = 1 : length(ensemble_complet)
            if ~isequal(ensemble_complet(h).elements, ENLEVE)
                ensemble_elements = ensemble_complet(h).elements;
                entre             = 0;
                for k = 1 : length(ensemble_elements)
                    %si une union, enleve
                    if isequal(ensemble_elements{k},ensemble_contrainte{1})
```

```
                        vecteur1             = ensemble_elements{k};
                        vecteur2             = ensemble_contrainte{1};
                        ensemble_elements{k} = setdiff(vecteur1, vecteur2);
                        entre                = 1;
                    end
                end

                if isequal(entre, 1)
                    j                    = 1;
                    ensemble_elements_new = [];
                    for k = 1 : length(ensemble_elements)
                        if ~isequal(ensemble_elements{k},[]);
                            ensemble_elements_new{j} = ensemble_elements{k};
                            j                        = j + 1;
                        end
                    end
                    ensemble_elements = [];
                    ensemble_elements = ensemble_elements_new;
                end
                ensemble_complet(h).elements = ensemble_elements;
            end
        end
    %sinon si elle est une intersection
    elseif length(ensemble_contrainte) == 1

        ensemble_produit = ensemble_complet;
        for t = 1 : length(ensemble_produit)
            ensemble        = ensemble_produit(t).elements;
            j               = 1;
            entre           = 1;
            nouvel_ensemble = {};
            for h = 1 : length(ensemble)
                for y = 1 : length(ensemble_contrainte)
                    if isequal(ensemble{h}, ensemble_contrainte{y})
                        ensemble{h} = [];
                        entre       = 0;
                    else
                        nouvel_ensemble{j} = ensemble{h};
                        j                  = j + 1;
                    end
                end
            end
            ensemble_produit(t).elements = nouvel_ensemble;
            ensemble_complet(t).elements = ensemble_produit(t).elements;
        end
    end
end

%enleve les vides
for r = 1 : length(ensemble_complet)
    ensemble1       = ensemble_complet(r).elements;
    j               = 1;
    nouvel_ensemble = [];
    for s = 1 : length(ensemble1)
        if ~isequal(ensemble1{s},[])
            nouvel_ensemble{j} = ensemble1{s};
            j                  = j + 1;
        end
    end
    ensemble_complet(r).elements = nouvel_ensemble;
end
```

```
%partie ci dessous combinera les lments identiques.
ensemble_complet = dedouble(ensemble_complet,2);
ensemble_complet = dedouble(ensemble_complet,1);
```

K Matlab code listing, file : ensemble.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function regrouping equal structure from matrix
%
% matrice: the matrix to regroup
%
% ensembleOut: outputs the structure with sets of regrouped matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ensembleOut] = ensemble(matrice)

%% for optimisation
%#inbounds
%#realonly

%% variables
ensembleOut = [];
[m,n]       = size(matrice);
j           = 1;

if ~(m < 2)
    if isequal(matrice(2,2).elements, [])
        u = 1;
        y = 1;
    else
        u = 2;
        y = 2;
    end
else
        u = 1;
        y = 1;
end

%% go through all sets of the matrix, put the equal ones togheter and sum
%% their mass
for g = u : m
    for h = y : n
        if isequal(g,u) & isequal(h,y) & ~isequal(matrice(g,h).elements,[])
            ensembleOut(j).elements = matrice(g,h).elements;
            ensembleOut(j).masses   = matrice(g,h).masses;
            j                       = j + 1;
        elseif ~isequal(matrice(g,h).elements,[])
            compris = 0;
            for f = 1 : length(ensembleOut)
                if isequal(matrice(g,h).elements, ensembleOut(f).elements)
                    ensembleOut(f).masses = ...
                          ensembleOut(f).masses + matrice(g,h).masses;
                    compris               = 1;
                end
            end
            if isequal(compris,0)
                 ensembleOut(j).elements = matrice(g,h).elements;
                 ensembleOut(j).masses   = matrice(g,h).masses;
```

```
                j                       = j + 1;
            end
        end
    end
end
```

L Matlab code listing, file : faire_contraire.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function that changes the sum of products in product of
%               sums with ADD and MULT
%
% info:         set that we want to modify
%
% ensembleOut:  once in product of sums and in same format as the input
% contraire:    only the first two information
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ensembleOut, contraire] = faire_contraire(info)

%% for optimisation
%#inbounds
%#realonly

%% variables
ensembleOut = [];
j           = 1;
f           = 1;
temp        = [];
flag        = 3;
contraire   = [];

%% puts the sets in product of sums
[temp, ignorance, nombre] = separation(info,2);
temp                      = produit_somme_complet(temp);
temp                      = depart(temp,2);

%% puts back the sets in one set
for g = 1 : length(nombre)
    debut           = 1;
    d               = 1;
    ensembleElement = {};
    for h = 1 : nombre(g)
        if isequal(debut,1)
            ensembleElement{d}    = [temp(f).elements];
            ensembleOut(j).masses = temp(f).masses;
            debut                 = 0;
        else
            ensembleElement{d}    = [temp(f).elements];
            ensembleOut(j).masses = [ensembleOut(j).masses, temp(f).masses];
        end
        f   = f + 1;
        d   = d + 1;
    end

    %% ensembleOut: output, once in product of sums
    ensembleOut(j).elements = ensembleElement;

    %% contraire: only the first two elements of output
    if j < 3
```

```
        contraire(j).elements = ensembleOut(j).elements;
        contraire(j).masses   = ensembleOut(j).masses;
    end
    j = j + 1;
end
```

M Matlab code listing, file : hybride.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:      function that executes the three steps of the DSmH
%
% info:             informations from the sources in product of sums
% contrainte:       contraints in sum of product
% contraire:        informations from sources in sum of products
%
% sorte:                indicates the type of fusion: dynamic ou static
% nombre_source:        number of source of evidence
% contraire_complet:    All the information in product of sum
%
% ensemble_complet:     final values (objects + masses)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ensemble_complet] = ...
        hybride(info,contrainte,contraire,sorte,nombre_source,contraire_complet)

%% variables
matrice_intersection = [];  matrice_union         = [];  matrice_monome       = [];
ensemble_step1       = [];  ensemble_step2        = [];  ensemble_step3       = [];
ensemble_complet     = [];  vecteur_singleton     = [];  contrainte_produit   = [];
ignorance            = [];  ensemble_complet_temp = [];

%% case static
if isequal(sorte,1)
    matrice_infos           = [];
    matrice_infos_contraire = [];
    for g = 1 : nombre_source
        [matrice_infos,ignorance,nombreElement]           = ...
                            separation_unique(info(g),matrice_infos);
        [matrice_infos_contraire,ignorance,nombreElement] = ...
               separation_unique(contraire_complet(g),matrice_infos_contraire);
    end

    %% compute the intersection matrix
    matrice_intersection = intersection_matrice(matrice_infos_contraire,2);
    matrice_intersection = somme_produit_complet(matrice_intersection);
    matrice_intersection = dedouble(matrice_intersection,2);
    matrice_intersection = ordre_grandeur(matrice_intersection,1);

    %% compute the union matrix
    matrice_intersection_contraire = intersection_matrice(matrice_infos,2);
    matrice_intersection_contraire = dedouble(matrice_intersection_contraire,2);

%% case dynamic
else

    %% Separates products of each objects, also computes total ignorance
    [matrice_monome,ignorance1,nombreElement]           = separation(info,1);
    [matrice_monome_contraire,ignorance2,nombreElement] = separation(contraire,1);
    ignorance                                           = [ignorance1];
```

```
    %% compute the union matrix
    matrice_intersection_contraire = intersection_matrice(matrice_monome,1);
    matrice_intersection_contraire = ...
                            ordre_grandeur(matrice_intersection_contraire,2);
    matrice_intersection_contraire = dedouble(matrice_intersection_contraire,2);

    %% compute the intersection matrix
    matrice_intersection = intersection_matrice(matrice_monome_contraire,1);
    matrice_intersection = somme_produit_complet(matrice_intersection);
    matrice_intersection = dedouble(matrice_intersection,2);
end

%% separates objects in contraints: will help compare the with intersection
if ~isempty(contrainte)
    [contrainte_separe, ignorance3, nombre] = separation(contrainte,3);
    contrainte_separe                       = tri(contrainte_separe,2);
end

%% compute S1, S2, S3

%% If there is no constraints, simply take S1
if isempty(contrainte)
    ensemble_complet = ensemble(matrice_intersection);

%% Otherwise, we have to go throught the three steps
else

    %% Go through intersection matrix, if objects = contraints, take union,
    %% if objects from union = contraints, take union of objects, if it's a
    %% contraints, take total ignorance.
    j     = 1;  source = 1;
    [m,n] = size(matrice_intersection);

    ss = 1:m; s = 1;
    gg = 1:n; g = 1;

    %% Go through each line (s) of the matrix process by accessing each
    %% objects, by column (g)

    while s ~= (length(ss)+1)
    while g ~= (length(gg)+1)

        %% take value from intersection matrix
        ensemble_step = matrice_intersection(s,g).elements;

        %% if the flag is not active, set it to '1'
        if ~(source > 10)
            source = 1;
        end

        %% Proceed if there is something at (s,g) matrix position
        if ~isequal(ensemble_step, [])
            intersection = 0;
            for h = 1 : length(contrainte_separe)

                %% If value from intersection matrix is equal to actual
                %% constraint and if it hasn't been equal to a previous
                %% constraint, OR, if the flag was active, then proceed to
                %% union matrix.
                if (isequal(contrainte_separe{h},ensemble_step) &...
                                    isequal(intersection,0)) | isequal(source,22)
```

```
intersection  = 1; union = 0;
ensemble_step = [];
ensemble_step =  matrice_intersection_contraire(s,g).elements;

%% if the flag is not active for the union of objects
%% or to total ignorance, set it to '2'
if ~(source > 22)
    source = 2;
end


for t = 1 : length(contrainte_separe)
    %% If value from union matrix is equal to actual
    %% constraint and if it hasn't been equal to a
    %% previous constraint, OR, if the flag was active,
    %% then proceed to union of objects calculation.
    if (isequal(contrainte_separe{t},ensemble_step)&...
                isequal(union,0))  | isequal(source,33)
        union = 1; subunion = 0;
        nouveau_vecteur = [];
        ensemble_step   = {};

            ensemble1 = matrice_monome(s,1).elements;
            ensemble2 = matrice_monome(1,g).elements;

            b = 1;
            for f = 1 : length(ensemble1)
            vecteur = ensemble1{f};
                for d = 1 : length(vecteur)
                    nouveau_vecteur{b} = [vecteur(d)];
                    b                  = b + 1;
                end
            end

            for f = 1 : length(ensemble2)
                vecteur = ensemble2{f};
                for d = 1 : length(vecteur)
                    nouveau_vecteur{b} = [vecteur(d)];
                    b                  = b + 1;
                end
            end

            %% remove repetition
            for f = 1 : length(nouveau_vecteur)
                for r = f + 1 : length(nouveau_vecteur)
          if isequal(nouveau_vecteur{f},nouveau_vecteur{r})
                    nouveau_vecteur{r} = [];
                end
            end
            end

            y = 1;
            for r = 1 : length(nouveau_vecteur)
                if ~isequal(nouveau_vecteur{r},[])
                    ensemble_step{y} = nouveau_vecteur{r};
                    y                = y + 1;
                end
            end

            %% ordering
            matrice               = [];
```

```
                    matrice(1,1).elements = ensemble_step;
                    matrice(1,1).masses   = 0;
                    matrice(2,2).elements = [];
                    matrice     = ordre_grandeur(matrice,2);
                    ensemble_step         = [];
                    ensemble_step         = matrice(1,1).elements;

                    %% if the flag is not active for ignorance
                    if ~(source > 33)
                        source = 3;
                    end

                for r = 1 : length(contrainte_separe)
                    %% If value from union of objects matrix is
                    %% equal to actual constraint and if it
                    %% hasn't been equal to previous constraint
                    %% OR, if the flag was active.
                    if (isequal(contrainte_separe{r}, ensemble_step)...
                        & isequal(subunion,0)) | isequal(source,44)
                        subunion = 1;
                        ensemble_step = {};
                        ensemble_step = ignorance;
                        source = 4;
                    end
                end

            end
        end

    end
end

ensemble_complet_temp = [];
ensemble_complet_temp(1).elements   = ensemble_step;
ensemble_complet_temp(1).masses = matrice_intersection(s,g).masses;

%% remove constraints of composed objects, if there is any
ensemble_step_temp = ...
   enlever_contrainte(ensemble_complet_temp,contrainte_separe);

%% once the contraints are all removed, check if the object are
%% empty. If not, increment output matrix position, if it is
%% empty, activate the flag following the position from where
%% the answer would have been taken and restart loop without
%% incrementing (s,g) intersection matrix position.
if ~isempty(ensemble_step_temp(1).elements)
    ensemble_step = [];
    ensemble_step = ensemble_step_temp(1).elements;
    ensemble_complet(j).elements = ensemble_step;
    ensemble_complet(j).masses  = ...
                            matrice_intersection(s,g).masses;
    ensemble_complet            = tri(ensemble_complet,1);
    j                           = j + 1;

else
    switch (source)
        %% CASE 4 is not used here. It's the case where there
        %% would be a constraint on total ignorance.
        case 1
           source = 22;
        case 2
```

```
                    source = 33;
                case 3
                    source = 44;
            end

            %% Will let the while loop repeat process for actual (s,g)
            g = g - 1;
        end


        end %% 'end' for the "if ~isequal(ensemble_step, [])" line

        %% move forward in the intersection matrix
        g = g + 1;
    end %g = 1 : n (columns of intersection matrix)

    %% move forward in the intersection matrix
    s = s + 1;
    g = 1;
    end %s = 1 : m (lines of intersection matrix)
end
g = 1; s = 1;

%% Sort the content of the output matrix
ensemble_complet = tri(ensemble_complet,1);

%% Filter the ouput matrix to merge equal cells
ensemble_complet = dedouble(ensemble_complet,1);
```

N Matlab code listing, file : intersection_matrice.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function that computes the intersection matrix and masses
%
% sorte:            type of fusion [static | dynamic]
% matrice_monome:   initial information, once separated by objects with ADD
% and MULT removed. vector represents products, a change of vector the sum
% includes only the first line and column of the matrix
%
% matrice_intersection: return the result of intersections
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [matrice_intersection] = intersection_matrice(matrice_monome,sorte)

%% case dynamic
if isequal(sorte,1)

    %% variables
    matrice_intersection = [];
    [m,n]                = size(matrice_monome);
    ensembleN            = {};
    ensembleM            = {};

    %% go through the first line and column, fill the intersection matrix
    for g = 2 : m
        ensembleM = matrice_monome(g,1).elements;
        for h = 2 : n
            ensembleN                          = matrice_monome(1,h).elements;
            matrice_intersection(g,h).elements = [ensembleN,ensembleM];
            matrice_intersection(g,h).masses   = ...
                  matrice_monome(g,1).masses * matrice_monome(1,h).masses;
        end
    end

    matrice_intersection = dedouble(matrice_intersection,2);
    matrice_intersection = ordre_grandeur(matrice_intersection,2);

%% case static
else

    %% variables
    matrice_intersection  = [];
    matrice_intermediaire = [];
    [m,n]                 = size(matrice_monome);
    ensembleN             = {};
    ensembleM             = {};
    j                     = 1;
    s                     = 1;

    %% fill the intersection matrix by multipliying all at once
    for g = 1 : n
        ensembleM = matrice_monome(1,g).elements;
        if ~isequal(ensembleM,[])
```

```
            for h = 1 : n
                    ensembleN = matrice_monome(2,h).elements;
                    if ~isequal(ensembleN,[])
                        matrice_intermediaire(j,s).elements = [ensembleN,ensembleM];
                        matrice_intermediaire(j,s).masses   = ...
                           matrice_monome(2,h).masses * matrice_monome(1,g).masses;
                        s                                   = s + 1;
                    end
            end
        end
    end

    [r,t] = size(matrice_intermediaire);
    s     = 1;
    for g = 3 : m
        for h = 1 : t
            ensembleM = matrice_intermediaire(1,h).elements;
            for u = 1 : n
                ensembleN = matrice_monome(g,u).elements;
                if ~isequal(ensembleN,[])
                    matrice_intersection(1,s).elements = [ensembleN,ensembleM];
                    matrice_intersection(1,s).masses   = ...
                  matrice_intermediaire(1,h).masses * matrice_monome(g,u).masses;
                    s                                  = s + 1;
                end
            end
        end
        matrice_intermediaire =  matrice_intersection;
        matrice_intersection  = [];
        [r,t] = size(matrice_intermediaire);
        s = 1;
    end

    matrice_intersection = matrice_intermediaire;
    matrice_intersection = dedouble(matrice_intersection,2);
end
```

O Matlab code listing, file : ordre_grandeur.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function that orders vectors
%
% matrice:      matrix in which we order the vectors in the sets
%
% matriceOut:   output ordered matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [matriceOut] = ordre_grandeur(matrice,sorte)

%% variables
[m,n]        = size(matrice);
ensemble     = {};
ensembleTemp = [];

%% case static
if isequal(sorte,1)
    u = 1;
    y = 1;

%% case dynamic
else
    essai = matrice(2,2).elements;
    if isempty(essai)
        u = 1;
        y = 1;
    else
        u = 2;
        y = 2;
    end
end

%% Order by size vector of sets of matrix
for g = u : m
    for h = y : n
        ensemble = matrice(g,h).elements;
        for f = 1 : length(ensemble)
            for k = f + 1 : length(ensemble)
                if length(ensemble{k}) < length(ensemble{f})
                    ensembleTemp = ensemble{f};
                    ensemble{f}  = ensemble{k};
                    ensemble{k}  = ensembleTemp;
                elseif isequal(length(ensemble{k}), length(ensemble{f}))
                    vecteur1 = ensemble{k};
                    vecteur2 = ensemble{f};
                    changer  = 0;
                    for t = 1 : length(vecteur1)
                        if (vecteur1(t) < vecteur2(t)) & isequal(changer,0)
                           ensembleTemp = ensemble{f};
                            ensemble{f} = ensemble{k};
                            ensemble{k} = ensembleTemp;
                            changer     = 1;
```

```
                        break;
                    end
                end
            end
        end
        matriceOut(g,h).elements = ensemble;
        matriceOut(g,h).masses   = matrice(g,h).masses;
    end
end
```

P Matlab code listing, file : plausibilite.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function that calculates plausibility
%
% I:            final information for which we want plausibility
% contrainte:   initial constraints
%
% plausibilite_complet: returns plausibility and masses
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function plausibilite_complet = plausibilite(I,contrainte)

%% for optimisation
%#inbounds
%#realonly

%% global variables
global ADD
global MULT
ADD  = -2;
MULT = -1;

%% variables
ensemble_complet   = {};
contrainte_compare = {};
info               = [];
matrice_monome     = [];
ignorance          = [];
ensemble_elements  = [];
vecteur1           = [];
vecteur2           = [];
nombreElement      = 0;
f                  = 1;
j                  = 1;
r                  = 1;

%% separates the objects, removes ADD and MULT
for g = 1 : length(I)
    if ~isempty(I(g).elements)
        ensemble_elements{f} = I(g).elements;
        vecteur2(f)          = I(g).masses;
        vecteur1(f)          = 1;
        f                    = f + 1;
    end
end
info(1).elements = ensemble_elements;
info(2).elements = ensemble_elements;
info(1).masses   = vecteur1;
info(2).masses   = vecteur2;
[matrice_monome,ignorance,nombreElement]           = separation(info,1);
[contraire_complet, contraire]                     = faire_contraire(info);
[matrice_monome_contraire,ignorance,nombreElement] = separation(contraire,1);
```

```
%% creates the intersection matrix
matrice_intersection = intersection_matrice(matrice_monome_contraire,1);
matrice_intersection = somme_produit_complet(matrice_intersection);
matrice_intersection = dedouble(matrice_intersection,2);

%% takes the contraint in sum of products, however, if there's none, do
%% nothing and put it all to '1'
entre = 0;
s     = 1;
for r = 1 : length(contrainte)
    if ~isempty(contrainte) & ~isempty(contrainte{r}) & isequal(entre,0)
        for g = 1 : length(contrainte)
            if ~isequal(contrainte{g},{})
                [contrainte_compare{s}, ignorance, nombre] = ...
                                        separation(contrainte{g},3);
                s                                          = s + 1;
            end
        end

        %% remove contraints on the intersection matrix
        [m,n] = size(matrice_intersection);
        for g = 2 : n
            ensemble_complet                           = [];
            matrice_intersection_trafique =  matrice_intersection(:,g);
            matrice_intersection_trafique(2,2).elements = [];
            ensemble_complet             = ensemble(matrice_intersection_trafique);
            ensemble_complet             = tri(ensemble_complet,1);
            ensemble_complet              = dedouble(ensemble_complet,1);
            for t = 1 : length(contrainte_compare)
                ensemble_complet = enlever_contrainte(ensemble_complet,...
                                                contrainte_compare{t});
            end
            resultat(j).masses = 0;
            for t = 1 : length(ensemble_complet)
                if ~isempty(ensemble_complet(t).elements)
                    resultat(j).masses = resultat(j).masses + ...
                                         ensemble_complet(t).masses;
                end
            end
            resultat(j).elements = matrice_monome(g,1).elements;
            j = j + 1;
        end
        entre = 1;

    %% if there's no constraints, put it all to '1',
    elseif isequal(length(contrainte),r) & isequal(entre,0)
        [m,n] = size(matrice_monome);
        for g = 1 : m
            resultat(j).elements = matrice_monome(g,1).elements;
            resultat(j).masses   = 1;
            j                    = j + 1;
        end
    end
end
plausibilite_complet = dedouble(resultat,1);
```

Q Matlab code listing, file : produit_somme_complet.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: function that converts input in product of sums
%
% ensemble_complet:     matrix in sum of products
%
% ensemble_produit:     matrix in product of sums
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ensemble_produit] = produit_somme_complet(ensemble_complet);

%% global variables
global ENLEVE
ENLEVE = {};

%% variables
ensemble_elements = {}; ensemble_produit  = {};
vecteur           = []; matrice           = [];
p                 = 1;  y                 = 1;

%% go through all sets, puts them in product of sums
for g = 1 : length(ensemble_complet)
    if ~isequal(ensemble_complet(g).elements, ENLEVE)
        ensemble_elements = ensemble_complet(g).elements;
        if (length(ensemble_elements) >= 2)
            i                           = 1;
            ensemble_produit(p).elements = {};
            changer                     = 0;

            if length(ensemble_elements) >= 3
                vecteur1 = ensemble_elements{1};
                vecteur2 = ensemble_elements{2};
                if ~(length(vecteur1) > 1 & length(vecteur2) > 1)
                    ensemble_produit(p).elements = ensemble_complet(g).elements;
                    ensemble_produit(p).masses   = ensemble_complet(g).masses;
                    p                           = p + 1;
                else
                    changer = 1 ;
                end
            else
                changer = 1;
            end

            if isequal(changer, 1)

                for k = 1 : length(ensemble_elements) - 1
                    if (k < 2)
                        if (k + 1) > length(ensemble_elements)
                            x = length(ensemble_elements);
                        else
                            x = k + 1;
                        end
```

```
            for w = k : x
                vecteur = ensemble_elements{w};
                j       = 1;
                for f = 1 : length(vecteur)
                    if isequal(length(vecteur),1)
                        ensembleN{j} = [vecteur];
                    else
                        ensembleN{j} = [vecteur(f)];
                        j            = j  + 1;
                    end
                end

                if isequal(i,1)
                    matrice(1,2).elements = ensembleN;
                    matrice(1,2).masses   = 0;
                    ensembleN             = {};
                    i                     = 2;
                elseif isequal(i,2)
                    matrice(2,1).elements = ensembleN;
                    matrice(2,1).masses   = 0;
                    ensembleN             = {};
                    i                     = 1;
                 end
            end

        elseif (k >= 2) & (length(ensemble_elements) > 2)
            w = k + 1;
            j = 1;
            vecteur = ensemble_elements{w};
            for f = 1 : length(vecteur)
                if isequal(length(vecteur),1)
                   ensembleN{j} = [vecteur];
                else
                   ensembleN{j} = [vecteur(f)];
                   j            = j  + 1;
                end
            end

            matrice(1,2).elements = ensemble_produit(p).elements;
            matrice(1,2).masses   = 0;

            matrice(2,1).elements = ensembleN;
            matrice(2,1).masses   = 0;
            ensembleN             = {};
        end

        resultat = union_matrice(matrice);
        [s,t]    = size(resultat);
        for r = 1 : s
            for d = 1 : t
                masse = resultat(r,d).masses;
                if isequal(masse, 0)
                    ensemble_produit(p).elements = ...
                                        resultat(r,d).elements;
                    ensemble_produit(p).masses   = ...
                                        ensemble_complet(g).masses;
                end
            end
        end

    end
```

```
                p = p + 1;
            end

        elseif isequal(length(ensemble_elements),1)
            for k = 1 : length(ensemble_elements)
                vecteur = ensemble_elements{k};
                j       = 1;
                for f = 1 : length(vecteur)
                    if isequal(length(vecteur),1)
                        ensembleN{j} = [vecteur];
                    else
                        ensembleN{j} = [vecteur(f)];
                        j            = j  + 1;
                    end
                end
            end

            ensemble_produit(p).elements = ensembleN;
            ensembleN                    = {};
            ensemble_produit(p).masses   = ensemble_complet(g).masses;
            p                            = p + 1;
        elseif ~isequal(ensemble_elements, [])
            ensemble_produit(p).elements = ensemble_complet(g).elements;
            ensemble_produit(p).masses   = ensemble_complet(g).masses;
            p                            = p + 1;
        end
    end
end
```

R Matlab code listing, file : separation.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  separates products in input data
%
% info:         information from sources (initial data)
% sorte:        type of separation
%
% retour:       separated data (products)
% ignorance:    total ignorance
% nombreElement:number of vectors in sets of each information
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [retour,ignorance_nouveau,nombreElement] = separation(info,sorte)

%% global variables
global ADD
global MULT
global SOURCE
ADD    = -2;
MULT   = -1;
SOURCE = 2;

%% variables
nouvelle_info     = []; %struc elements: set of vector
ensemble_monome   = []; %cell (1,1) of matrix is empty
matrice_monome    = []; %cell (1,1) of matrix is empty
retour            = [];
ignorance_nouveau = [];

%% takes each elements of each sources and separates the products

[m,n] = size(info);

%% computes the number of elements
if ~isequal(sorte,3)
    for g = 1 : n
        nombreElement(g) = length(info(g).elements);
    end
else
    nombreElement(1) = 1;
end

%% case dynamic or two sources
if isequal(sorte,1)

    %% variables
    ligne          = 1;
    colonne        = 2;
    ignorance      = [];

    %% go through each sources
    for g = 1 : n
        ensemble      = info(g).elements;
```

```
            vecteur_masse = info(g).masses;
            if isequal(g,SOURCE)
                colonne = 1;
                ligne   = 2;
            end

            %% go through each set of elements
            for h = 1 : length(ensemble)
                vecteur         = ensemble{h};
                nouveau_vecteur = [];
                nouvel_ensemble = {};
                k               = 1;

                %% go through each element of the vector
                %%     to separate the products and sums
                for j = 1 : length(vecteur)
                    if ~isequal(vecteur(j), ADD)
                        if ~isequal(nouveau_vecteur, []) & ~isequal(vecteur(j), MULT)
                            nouveau_vecteur = [nouveau_vecteur, vecteur(j)];
                            if isequal(j,length(vecteur))
                               nouvel_ensemble{k} = nouveau_vecteur;
                               ignorance          = [ignorance, nouveau_vecteur];
                            end
                        elseif ~isequal(vecteur(j), MULT)
                            nouveau_vecteur = [vecteur(j)];
                            if isequal(j,length(vecteur))
                               nouvel_ensemble{k} = nouveau_vecteur;
                               ignorance          = [ignorance, nouveau_vecteur];
                            end
                        end
                    else
                        nouvel_ensemble{k} = nouveau_vecteur;
                        ignorance          = [ignorance, nouveau_vecteur];
                        nouveau_vecteur    = [];
                        k                  = k + 1;
                    end
                end

                nouvelle_info(g,h).elements = nouvel_ensemble;
                nouvelle_info(g,h).masses   = vecteur_masse(h);
                if isequal(g,1)
                    matrice_monome(ligne,colonne).elements = nouvel_ensemble;
                    matrice_monome(ligne,colonne).masses   = vecteur_masse(h);
                    colonne                                = colonne + 1;
                elseif isequal(g,2)
                    matrice_monome(ligne,colonne).elements = nouvel_ensemble;
                    matrice_monome(ligne,colonne).masses   = vecteur_masse(h);
                    ligne                                  = ligne + 1;
                end
            end
        end

        ignorance = unique(ignorance);
        for r = 1 : length(ignorance)
            ignorance_nouveau{r} = ignorance(r);
        end
        retour = matrice_monome;

    %% case static
    elseif isequal(sorte,2)
```

```
    %% variables
    f              = 1;

    %% go through each sources
    for g = 1 : n
        ensemble      = info(g).elements;
        vecteur_masse = info(g).masses;

        %% go through each set of elements
        for h = 1 : length(ensemble)
            vecteur         = ensemble{h};
            nouveau_vecteur = [];
            nouvel_ensemble = {};
            k               = 1;


            %% go through each element of the vector
            %%     to separate the products and sums
            for j = 1 : length(vecteur)
                if ~isequal(vecteur(j), ADD)
                    if ~isequal(nouveau_vecteur, []) & ~isequal(vecteur(j), MULT)
                        nouveau_vecteur = [nouveau_vecteur, vecteur(j)];
                        if isequal(j,length(vecteur))
                           nouvel_ensemble{k} = nouveau_vecteur;
                        end
                    elseif ~isequal(vecteur(j), MULT)
                        nouveau_vecteur = [vecteur(j)];
                        if isequal(j,length(vecteur))
                           nouvel_ensemble{k} = nouveau_vecteur;
                        end
                    end
                else
                    nouvel_ensemble{k} = nouveau_vecteur;
                    nouveau_vecteur    = [];
                    k                  = k + 1;
                end
            end
            ensemble_monome(f).elements = nouvel_ensemble;
            ensemble_monome(f).masses  = vecteur_masse(h);
            f                          = f + 1;
        end
    end
    ignorance = [];
    retour    = ensemble_monome;

%% case contraint
elseif isequal(sorte,3)
    for g = 1 : length(info)
        vecteur         = info{g};
        nouveau_vecteur = [];
        nouvel_ensemble = {};
        k               = 1;
        for h = 1 : length(vecteur)
            if ~isequal(vecteur(h), ADD)
                 if ~isequal(nouveau_vecteur, []) & ~isequal(vecteur(h), MULT)
                     nouveau_vecteur = [nouveau_vecteur, vecteur(h)];
                     if isequal(h,length(vecteur))
                        nouvel_ensemble{k} = nouveau_vecteur;
                     end
                 elseif ~isequal(vecteur(h), MULT)
                     nouveau_vecteur = [vecteur(h)];
```

```
                    if isequal(h,length(vecteur))
                       nouvel_ensemble{k} = nouveau_vecteur;
                    end
               end
          else
               nouvel_ensemble{k} = nouveau_vecteur;
               nouveau_vecteur    = [];
               k                  = k + 1;
          end
       end
       nouvelle_contrainte{g} = nouvel_ensemble;
    end
    ignorance = [];
    retour    = nouvelle_contrainte;
end
```

S Matlab code listing, file : separation_unique.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  separates products in input data, one info. at a time
%
% info:         information from sources (initial data)
% sorte:        type of separation
%
% matrice_monome:   separated data (products)
% ignorance:        total ignorance
% nombreElement:    number of vectors in sets of each information
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [matrice_monome,ignorance,nombreElement] = ...
                                        separation_unique(info,matrice_monome)

%% for optimisation
%#inbounds
%#realonly

%% global variables
global ADD
global MULT
global SOURCE
ADD    = -2;
MULT   = -1;
SOURCE = 2;

%% variables
nouvelle_info  = []; %struc elements: set of vector
ignorance      = [];

if isequal(matrice_monome, [])
    ligne   = 1;
    colonne = 1;
else
    [m,n]   = size(matrice_monome);
    ligne   = m + 1;
    colonne = 1;
end

%% takes each elements of each sources and separates the products

[m,n] = size(info);

%% computes the number of elements
for g = 1 : n
    nombreElement(g) = length(info(g).elements);
end

%% go through each sources
for g = 1 : n
    ensemble      = info(g).elements;
    vecteur_masse = info(g).masses;
```

```
    %% go through each set of elements
    for h = 1 : length(ensemble)
        vecteur         = ensemble{h};
        nouveau_vecteur = [];
        nouvel_ensemble = {};
        k               = 1;

        %% go through each elements of the vector
        %% separates the products and sums
        for j = 1 : length(vecteur)
            if ~isequal(vecteur(j), ADD)
                if ~isequal(nouveau_vecteur, []) & ~isequal(vecteur(j), MULT)
                    nouveau_vecteur = [nouveau_vecteur, vecteur(j)];
                    if isequal(j,length(vecteur))
                       nouvel_ensemble{k} = nouveau_vecteur;
                       ignorance          = [ignorance, nouveau_vecteur];
                    end
                elseif ~isequal(vecteur(j), MULT)
                    nouveau_vecteur = [vecteur(j)];
                    if isequal(j,length(vecteur))
                       nouvel_ensemble{k} = nouveau_vecteur;
                       ignorance          = [ignorance, nouveau_vecteur];
                    end
                end
            else
                nouvel_ensemble{k} = nouveau_vecteur;
                ignorance          = [ignorance, nouveau_vecteur];
                nouveau_vecteur    = [];
                k                  = k + 1;
            end
        end

        nouvelle_info(g,h).elements            = nouvel_ensemble;
        nouvelle_info(g,h).masses              = vecteur_masse(h);
        matrice_monome(ligne,colonne).elements = nouvel_ensemble;
        matrice_monome(ligne,colonne).masses   = vecteur_masse(h);
        colonne                                = colonne + 1;
    end
end

ignorance = unique(ignorance);
```

T Matlab code listing, file : somme_produit_complet.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: function that converts input in sum of products
%
% matrice_contraire:     matrix in product of sums
%
% matrice_complet:     matrix in sum of products
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [matrice_complet] = somme_produit_complet(matrice_contraire);

%% for optimisation
%#inbounds
%#realonly

%% variables
ensemble_elements = {};
vecteur           = [];
matrice           = [];
matrice_complet   = [];
p                 = 1;
ensembleN         = {};
[m,n]             = size(matrice_contraire);

if ~isempty(matrice_contraire(1,1).elements)
 u = 1;
 v = 1;
else
  u = 2;
  v = 2;
end

%% go through the sets and puts them in sum of product
for g = u : m
    for t = v : n
        ensemble_elements = matrice_contraire(g,t).elements;
        if ~isequal(ensemble_elements, {})
            matrice_complet(g,t).elements = {};
            matrice_complet(g,t).masses   = 0;
            ensembleN                     = {};
            if isequal(length(ensemble_elements), 1)
                vecteur      = ensemble_elements{1};
                j            = 1;
                ensembleN{j} = [];
                for f = 1 : length(vecteur)
                    ensembleN{j} = [vecteur(f)];
                    j            = j + 1;
                end
                matrice_complet(g,t).elements = ensembleN;
                matrice_complet(g,t).masses   = matrice_contraire(g,t).masses;

            elseif length(ensemble_elements) >= 2
                matrice_complet(g,t).elements = [];
```

```
changer                      = 0;

if length(ensemble_elements) >= 3
    vecteur1 = ensemble_elements{1};
    vecteur2 = ensemble_elements{2};
    %file produit_somme_complet.m needed an '~' for the IF
    %here to work as it should be.
    if (length(vecteur1) > 1 & length(vecteur2) > 1)
        matrice_complet(g,t).elements = ...
                                matrice_contraire(g,t).elements;
        matrice_complet(g,t).masses   = ...
                                matrice_contraire(g,t).masses;
    else
        changer = 1 ;
    end
else
    changer = 1;
end

if isequal(changer,1);
    matrice_complet(g,t).elements = {};
    i                              = 1;

    for k = 1 : length(ensemble_elements) - 1
    if (k < 2)
        if (k + 1) > length(ensemble_elements)
            x = length(ensemble_elements);
        else
            x = k + 1;
        end

        for w = k : x
            vecteur = ensemble_elements{w};
            j       = 1;
            for f = 1 : length(vecteur)
                if isequal(length(vecteur),1)
                    ensembleN{j} = [vecteur];
                else
                    ensembleN{j} = [vecteur(f)];
                    j            = j  + 1;
                end
            end

            if isequal(i,1)
                matrice(1,2).elements = ensembleN;
                matrice(1,2).masses   = 0;
                ensembleN             = {};
                i                     = 2;
            elseif isequal(i,2)
                matrice(2,1).elements = ensembleN;
                matrice(2,1).masses   = 0;
                ensembleN             = {};
                i                     = 1;
             end
        end
    elseif (k >= 2) & (length(ensemble_elements) > 2)
        w = k + 1;
        j = 1;
        vecteur = ensemble_elements{w};
        for f = 1 : length(vecteur)
            if isequal(length(vecteur),1)
```

```
                        ensembleN{j} = [vecteur];
                    else
                        ensembleN{j} = [vecteur(f)];
                        j            = j  + 1;
                    end
                end

                matrice(1,2).elements = matrice_complet(g,t).elements;
                matrice(1,2).masses   = 0;

                matrice(2,1).elements = ensembleN;
                matrice(2,1).masses   = 0;
                ensembleN             = {};
            end

                matrice               = ordre_grandeur(matrice,2);
                resultat              = union_matrice(matrice);

                matrice(2,1).elements = {};
                matrice(1,2).elements = {};

                [s,b] = size(resultat);
                for r = 1 : s
                for d = 1 : b
                    masse = resultat(r,d).masses;
                    if isequal(masse, 0)
                        matrice_complet(g,t).elements = ...
                                        resultat(r,d).elements;
                        matrice_complet(g,t).masses   = ...
                                       matrice_contraire(g,t).masses;
                    end
                end
                end
            end
        end
        elseif ~isequal(ensemble_elements, [])
            matrice_complet(g,t).elements = matrice_contraire(g,t).elements;
            matrice_complet(g,t).masses   = matrice_contraire(g,t).masses;
        end
    end
end
end

if (g >= 2) & (t >= 2)
    matrice_complet = ordre_grandeur(matrice_complet,2);
end
```

U Matlab code listing, file : tri.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function that sorts the elements
%
% matrice:      matrix to sort, can be a set
% sorte:        type of input [matrix | set]
%
% retour:       matrix, or set, once the elements are sorted
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [retour] = tri(matrice,sorte)

%% for optimisation
%#inbounds
%#realonly

%% case matrix
if isequal(sorte,1)

    %% variables
    [m,n]         = size(matrice);
    ensemble_temp = [];

    if m > 1
        u = 2;
        v = 2;
    else
        u = 1;
        v = 1;
    end

    %% go through each elements of the matrix, sort them
    for h = u : m
        for g = v : n
            ensemble = matrice(h,g).elements;
            for f = 1 : length(ensemble)
                for k = f + 1 : length(ensemble)

                    %% if they are the same length, look at each number, at
                    %% order them
                    if isequal(length(ensemble{f}),length(ensemble{k}))
                        if (ensemble{f} > ensemble{k})
                            ensemble_temp = ensemble{f};
                            ensemble{f}   = ensemble{k};
                            ensemble{k}   = ensemble_temp;
                        end
                    else

                        %% ifnot the same length, put at first, the smaller
                        if length(ensemble{f}) > length(ensemble{k})
                            ensemble_temp = ensemble{f};
                            ensemble{f}   = ensemble{k};
                            ensemble{k}   = ensemble_temp;
```

```
                    end
                end
            end
        end
        matriceOut(h,g).elements = ensemble;
        matriceOut(h,g).masses   = matrice(h,g).masses;
    end
end

retour = matriceOut;

%% case set
else

    %% variables
    ensemble_temp = [];

    %% go through each elements of the set, sort them
    for h = 1 : length(matrice)
        ensemble_tri = matrice{h};
        for f = 1 : length(ensemble_tri)
            for k = f + 1 : length(ensemble_tri)
                if isequal(length(ensemble_tri{f}),length(ensemble_tri{k}))
                    if (ensemble_tri{f} > ensemble_tri{k})
                        ensemble_temp   = ensemble_tri{f};
                        ensemble_tri{f} = ensemble_tri{k};
                        ensemble_tri{k} = ensemble_temp;
                    end
                else
                    if length(ensemble_tri{f}) > length(ensemble_tri{k})
                        ensemble_temp   = ensemble_tri{f};
                        ensemble_tri{f} = ensemble_tri{k};
                        ensemble_tri{k} = ensemble_temp;
                    end
                end
            end
        end
        ensembleOut{h} = ensemble_tri;
    end

    retour = ensembleOut;
end
```

V Matlab code listing, file : union_matrice.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Property of the Government of Canada, DRDC Valcartier
% Authors : M.-L. Gagnon and P. Djiknavorian
%
% Under supervision of : Dominic Grenier
% Location      : Laval University, Quebec
% Last update   : 20 may 2006
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:  function that computes the union matrix and masses
%
% matrice_monome: objects and masses once separated, on the 1st line/column
%
% matrice_union : returns the result of the union and masses
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [matrice_union] = union_matrice(matrice_monome)

%% for optimisation
%#inbounds
%#realonly

%variables
matrice_union = [];
[m,n]         = size(matrice_monome);
ensembleN     = {};
ensembleM     = {};
vecteurN      = [];
vecteurM      = [];
ensemble1     = [];
ensemble2     = [];

%% go through the 1st line and column, fill the union matrix
for g = 2 : n
    ensembleN = matrice_monome(1,g).elements;
    if ~isequal(ensembleN,{})
        for h = 2 : m
            ensembleM = matrice_monome(h,1).elements;
            if ~isequal(ensembleM,{})
                if isequal(ensembleN, ensembleM)
                    matrice_union(h,g).elements = ensembleN;
                    matrice_union(h,g).masses   = matrice_monome(1,g).masses *...
                                                  matrice_monome(h,1).masses;
                else

                    %% put the identical ones (from same line) togheter
                    [ensembleM,ensembleN] = bon_ordre(ensembleM,ensembleN);

                    %% verifies which one is the higher
                    if length(ensembleM) >= length(ensembleN)
                        ensemble1 = ensembleN;
                        ensemble2 = ensembleM;
                    else
                        ensemble1 = ensembleM;
                        ensemble2 = ensembleN;
                    end
                end

                %% fill the union matrix
                nouvel_ensemble = {};
```

```
                j               = 1;
                for t = 1 : length(ensemble1)
                    for s = 1 : length(ensemble2)
                        if t <= length(ensemble2)
                            if isequal(ensemble2{s},ensemble1{t})
                                nouvel_ensemble{j} = [ensemble1{t}];
                            else
                                vecteur = [ensemble2{s},ensemble1{t}];
                                nouvel_ensemble{j} = unique(vecteur);
                            end
                        else
                            if isequal(ensemble1{length(ensemble2)},ensemble1{t})
                                nouvel_ensemble{j} = [ensemble1{t}];
                            else
                                vecteur = ...
                                     [ensemble1{length(ensemble2)},ensemble1{t}];
                                nouvel_ensemble{j} = unique(vecteur);
                            end
                        end
                        j = j + 1;
                    end
                end
                matrice_union(h,g).elements = nouvel_ensemble;
                matrice_union(h,g).masses   = matrice_monome(1,g).masses *...
                                              matrice_monome(h,1).masses;
            end
        end
    end
end
matrice_union = ordre_grandeur(matrice_union,2);
```

Bibliographie

[1] Ayyub, B.M., Klir, G.J., *Uncertainty modeling and Analysis in Engineering and the Sciences*, Chapman & Hall - CRC, USA, 2006.

[2] Bayes, T., *An Essay Toward Solving a Problem in the Doctrine of Chances*, volume 53, 1763. Reprinted in *Facsimiles of two papers by Bayes*, Hafner Publishing Company, New York, 1963.

[3] Liu, C., *A general measure of uncertainty-based information*, Thèse de doctorat, Université Laval, 2004.

[4] Delmotte, F., Dubois, L., Desodt, A.M., Borne, P., *Trust in uncertainty theories*, *Information and system engineering*, Vol. 1, no. 3-4, pp. 303-314, 1995.

[5] Dempster, A., *Upper and lower probabilities induced by multivalued mapping*, *Annals of Mathematical Statistics*, AMS-38, pp. 325-339, 1967.

[6] Djiknavorian, P., Grenier, D., *Reducing DSmT hybrid rule complexity through optimisation of the calculation algorithm*, Chapitre de [26], 2006.

[7] Djiknavorian, P., Grenier, D., Valin, P., *Analysis of information fusion combining rules under the DSm theory using ESM input*, ISIF Fusion 2007 Conference, Quebec, Canada, July 2007.

[8] Djiknavorian, P., Grenier, D., Valin, P., *Extension of the ACR combining rule to DSmT*, [en cours de rédaction]

[9] Djiknavorian, P., Grenier, D., Valin, P., *Monte-Carlo study of combining rules under the DSmT*, [en cours de rédaction]

[10] Dubois, D., Prade, H., *Fuzzy Sets and Systems : Theory and Applications*, Academic Press, 1980.

[11] Dubois, D., Prade, H., *Possibility Theory - An approach to the Computerized Processing of Uncertainty*, Academic Press, 1988.

[12] Florea, M.C., *Fusion d'informations imparfaites dans le cadre unificateur des ensembles aléatoires - Application à l'identification de cibles*, Mémoire de maîtrise, Université Laval, 2003.

[13] Florea, M.C., *et al. Adaptive Combination Rule and Proportional Conflict Redistribution Rule for Information Fusion*, COGIS 2006 Conference, Paris, France, 2006.

[14] Gagnon, M.-L., Grenier, D., *Rapport : Fusion de donnes, été 2005*, Rapport technique, Université Laval, 2005.

[15] Haenni, R., *Shedding new light on Zadeh's criticism of Dempster's rule of combination*, ISIF Fusion 2005 Conference, Philadelphia, USA, July 2005.

[16] Hall, D.L., Llinas, J., (Editors) *Handbook of Multisensor Data Fusion*, CRC Press LLC, 2001.

[17] Hall, D.L., McMullen, S.A.H., *Mathematical Techniques in Multisensor Data Fusion*, Second Edition, Artech House, 2004.

[18] Inagaki, T., *Interdependence between safety-control policy and multiple-sensor schemes via Dempster-Shafer theory*, IEEE Trans. on reliability, vol. 40, no. 2, pp. 182-188, 1991.

[19] Klein, L.A., *Sensor and Data Fusion : A Tool for Information Assessment and Decision Making*, SPIE Press Monograph, Vol. PM138, 2004.

[20] Mahler, R.P.S., *Statistical Multisource multitarget information fusion*, Artech House, 2007.

[21] Merrill, S., Grofman, B., *A unified theory of voting* Cambridge, 1999.

[22] Sentz, K., Ferson, S., *Combination of Evidence in Dempster-Shafer Theory*, Sandia National Laboratory - Epistemic Uncertainty Project, Tech. Rep. SAND2002-0835, 2002.

[23] Shafer, G., *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, New Jersey, 1976.

[24] Smarandache, F., *Unification of Fusion Theories*, NATO Advanced Study Institute, Albena, Bulgaria, 16-27 May 2005.
`http://www.gallup.unm.edu/~smarandache/UnificationFusionTheories.pdf`

[25] Smarandache, F., Dezert, J. (Editors), *Advances and Applications of DSmT for Information Fusion*, (Collected Works), vol. 1, USA : American Research Press, Rehoboth, 2004.
`http://www.gallup.unm.edu/~smarandache/DSmT-book1.pdf`

[26] Smarandache, F., Dezert, J. (Editors), *Advances and Applications of DSmT for Information Fusion*, (Collected Works), vol. 2, USA : American Research Press, Rehoboth, 2006.
`http://www.gallup.unm.edu/~smarandache/DSmT-book2.pdf`

[27] Smets, Ph., Kennes, R., *The transferable belief model*, *Artificial Intelligence*, 66(2), pp. 191-234, 1994.

[28] Smets, Ph., *Data Fusion in the Transferabe Belief Model*, Proc. of the 3rd International Conference on Information Fusion, Paris, July 10-13, 2000.

[29] Steinberg, A.N., Bowman, C.L., White, Jr., F.E., *Revisions to the JDL Data Fusion Model*, Proc. 3rd NATO/IRIS Conf., Quebec City, Canada, 1998.

[30] Valin, P., Bossé, É., Jouan, A., *Information fusion concepts for airborne maritime surveillance and C2 operations*, DRDC Valcartier TM 2004-281, 2006.

[31] Valin, P., Boily, D., *Truncated Dempster-Shafer Optimization and Benchmarking*, 'Sensor Fusion : Architectures, Algorithms, and Applications IV', SPIE Aerosense 2000, Orlando, Florida, April 24-28, pp. 237-246, 2000.

[32] Walley, P., *Statistical Reasoning with Imprecise Probabilities*, Chapman & Hall, 1991.

[33] Waltz, E., Llinas, J., (Editors) *Multisensor Data Fusion*, Artech House Publishers, 1990.

[34] Weisstein, E. W., *Antichain. From MathWorld*, A Wolfram Web Resource. 2007. `http://mathworld.wolfram.com/Antichain.html`

[35] White, Jr., F.E., *Data Fusion Lexicon*, Joint Directors of Laboratories, Technical Panel for C^3, Data Fusion Sub-Panel, Naval Ocean Systems Center, San Diego, 1987.

[36] Yager, R.R., *On the Dempster-Shafer framework and new combination rules*, Information Sciences, Vol. 41, pp.43-138, 1987.

[37] Yager, R.R., *Hedging in the Combination of Evidence*, Journal of Information and Optimization Science, Vol. 4, no.1, pp. 73-81, 1983.

[38] Zadeh, L.A., *Fuzzy sets*, *Information and Control*, 8, pp. 338-353, 1965.

[39] Zadeh, L.A., *Fuzzy sets as a basis for a theory of possibility*, *Fuzzy Sets and Systems*, 1, pp. 3-28, 1978.

[40] Zadeh, L.A., *Review of Shafer's a mathematical theory of evidence*, *AI Mag.*, vol. 5, pp. 81-83., 1984.