Hybridation de l'Answer Set Programming et de la théorie de Dempster Shafer

Serge Sonfack Sounchio, Laurent Geneste, Bernard Kamsu Foguem

Hybridation de l'Answer Set Programming et de la théorie de Dempster Shafer

S. SONFACK SOUNCHIO[1], L. GENESTE[1], B. KAMSU FOGUEM[1]

[1] Université de Toulouse, Laboratoire Génie de Production

10 juin 2021

Résumé

Au cours des processus d'expertise, qui permettent d'explorer les différentes solutions d'un problème, il est utile de raisonner en intégrant à la fois un raisonnement non-monotone et l'incertitude. Nous proposons pour cela une approche hybridant la théorie de Dempster Shafer (pour l'incertitude) et l'approche Answer Set Programming (pour le raisonnement non monotone). Un choix peut ainsi être effectué, parmi les ensembles solution du programme logique, en utilisant conjointement une mesure de croyance et le niveau d'incohérence résultant de la base de connaissances.

Mots-clés

Représentation de la connaissance, Answer Set Programming, théorie des fonctions de croyance, incertitude.

Abstract

In practical situations it is useful to be able to reason about uncertain knowledge within a non-monotonic logic. This is, the case of expertise processes which finds all possible solutions of a cause. We propose for this purpose an approach that aims at combining the theory of belief functions and Answer Set Programming (ASP) by allowing decision-making with the generalized ordered weighted average.

Keywords

Knowledge représentation, Answer Set Programming, belief function theory, uncertainty.

1 Introduction

Les démarches d'expertise se caractérisent par des processus fortement exploratoires fondés sur un raisonnement à base d'hypothèses (intégrant donc de l'incertitude) et la manipulation de connaissances expertes. Elles sont mises en œuvre dans des contextes très variés comme la conception de systèmes, la résolution de problèmes industriels ou encore l'expertise médico-légale. Un référentiel normatif sur les démarches d'expertise a été défini, d'abord en France avec la norme NF X50-110 [19] "Qualité des activités d'expertise", puis au niveau européen avec la norme CSN EN 16775 "Activités d'expertise - Exigences générales pour les services d'expertise". Cependant, si ces normes définissent des principes et une description des processus d'expertise, elles ne fournissent pas d'outils pour modéliser les problèmes à expertiser et faciliter la prise de décision en ce qui concerne le choix des solutions du processus d'expertise.

L'espace de prise de décision humaine dans les processus d'expertise fait apparaître principalement deux types de défauts selon [29] : (1) les défauts qualitatifs résultant des limites de la représentation du monde réel ; (2) les défauts numériques liés à l'incertitude du dit monde.

Ainsi, prendre en compte ces deux inconvénients pour la prise de décision permettra de l'améliorer et de réduire les risques ou conséquences des décisions dans certains domaines [18] [22].

C'est dans cette optique de considérer ces défauts que nous proposons dans cet article une méthode permettant de déterminer une valeur de prise de décision à partir d'une représentation qualitative et quantitative de la connaissance d'un domaine donné. Ainsi, pour la représentation qualitative, nous nous sommes penchés sur la programmation par ensemble réponses (Answer Set Programming, ASP), qui au-delà de son expressivité par rapport aux langages basés simplement sur la logique classique, est non monotone et modélise bien le raisonnement intégrant le bon sens [5, 1], ce qui la rend appropriée pour raisonner avec une connaissance incomplète d'un domaine.

En outre, il est important de tenir compte de l'incertitude quantitative dans cette représentation, car selon [4], elle aide à surmonter certaines limites de l'ASP, qui ne peut exprimer de manière naturelle les croyances des littéraux, ni leur véracité quantifiée et même leur manque de connaissance. En général, l'ASP classique ne peut pas exprimer intuitivement la quantité d'incertitude de sa sémantique.

Bien que des approches pour apporter la quantification de l'incertitude en représentation de la connaissance à base de logique aient été proposées, le problème reste d'actualité [7]. Il est à noter que la plupart des méthodologies reposent uniquement sur les théories des probabilités ou des possibilités appliquées aux règles logiques [3, 7] [11]. Tandis que nous proposons une approche utilisant d'une part la théorie de Dempster Shafer qui est une généralisation des théories précédentes [21] et d'autre part la mesure de l'inconsistance de la connaissance représentée en ASP.

De cette façon, nous cherchons à repousser les limites des ensembles solutions (Answer Sets) en ce qui concerne la gestion des connaissances incertaines[34].

Le reste de ce document est structuré de la manière suivante : en section 2, nous présentons la programmation par ensemble réponses et la théorie de Dempster Shafer. En section 3 nous décrivons l'approche proposée pour calculer une valeur de décision, en combinant la programmation par ensemble réponses et la théorie de Dempster Shafer. Nous terminerons cette section par un exemple d'application illustrant la méthode.

2 Fondamentaux

2.1 Programmation par ensemble réponses (Answer Set Programming, ASP)

L'ASP est une approche de représentation de la connaissance basée sur un paradigme déclaratif reposant sur la logique de premier ordre. Elle utilise un solveur basé sur la recherche de modèles stables, semblable à celui de la programmation par contraintes [12, 27, 8]. Bien qu'utilisant une syntaxe similaire au langage Prolog, ce langage est adapté pour la résolution de problèmes d'ordre multiples (web sémantique, planification, théorie des graphes, bio-informatique, configuration) et tire ses racines du raisonnement non monotone [13]. La non-monotonicité de l'ASP est obtenue par une forme de négation, appelée *négation comme échec* qui peut s'assimiler au raisonnement par défaut [14]. Cela en fait un choix approprié pour le raisonnement de bon sens, proche du raisonnement humain.

2.1.1 Syntaxe ASP

La programmation en ensemble réponses peut se reposer aussi bien sur la logique propositionnelle que sur la logique du premier ordre [24]. De plus elle dispose d'extensions avec des règles de choix, de pondération ou de cardinalité [28].
Un programme ASP est constitué d'un ensemble de règles (r) se présentant sous la forme suivante :
$a \leftarrow b_1, \ldots, b_n, notb_{n+1}, \ldots, notb_m$, $0 \leq n \leq m$ où a et b_i sont atomes.
a est appelé la *tête* de la règle et est notée par $tete(r)$
$\{b_1, \ldots, b_n, b_{n+1}, \ldots, b_m\}$ forment le corps de celle-ci et se note $corps(r)$.
$corps(r)$ peut être divisé en deux parties :

— Les *littéraux positifs* $corps^+(r) = \{b_1, \ldots, b_n\}$
— Les *littéraux négatifs* $corps^-(r) = \{b_{n+1}, \ldots, b_m\}$

Il est possible d'avoir des règles sans corps ($a \leftarrow$, c'est-à-dire $corps(r) = \emptyset$) aussi appelés *fait*.
Si $corp^-(r) = \emptyset$, c'est-à-dire si la règle ne dispose pas de littéraux négatifs, on parle de *règle définie* (*definite rule*). Ainsi lorsqu'un programme est constitué uniquement de règles définies, il est appelé *programme défini* sinon on parle de *programme normal*
La négation *not* que nous retrouvons dans une règle est différente de la négation classique ($\neg$), que l'on retrouve en logique classique. En effet, dans le cadre de la programmation par ensemble réponses, cette négation est appelée : *négation par défaut* et a le sens de : "*on ne pense pas que*". C'est cette sémantique qui permet de faire un raisonnement par défaut.
Par définition, un programme de la programmation par ensemble réponses est constituée d'un ensemble de règles comme nous l'avons défini ci-dessus, mais dispose aussi d'instructions comme *règles de choix, règles pondérées, règles de cardinalité* qui ont pour rôle d'ajouter des contraintes sur l'ensemble réponse (*Answer Set*).

2.1.2 Sémantique de ASP

La signification des programmes est donnée par la sémantique du *modèle stable*, qui définit un ensemble d'atomes satisfaisant toutes les règles du programme. Cet ensemble d'atomes est appelé *ensemble réponse* (*Answer Set*) et correspond au *point fixe* de l'*opérateur de conséquence immédiate* (*single step operator*) 2.1.2
Un ensemble d'atomes est obtenu par extension M est un ensemble de réponses d'un programme ASP s'il satisfait aux conditions suivantes : [15, 20] :

— Si le programme ASP est un programme défini, c'est-à-dire constitué uniquement de règles définies, alors M est un ensemble minimal d'atomes qui satisfait à toutes les règles de ce programme ASP.
— Si le programme ASP est un programme normal (c'est-à-dire non défini), alors M coïncide avec l'ensemble réponse de la réduction connue sous le nom de *réduction Gelfond-Lifschitz* du programme à sa forme définie.

Lorsqu'un programme ASP a un ensemble réponse (*Answer Set*), on dit qu'il est **consistant** et **inconsistant** dans le cas contraire.
La **réduction Gelfond-Lifschitz** est un processus de transformation d'un programme normal en un programme défini (sans règle négative), par rapport à un ensemble d'atomes. Elle est basée sur la forme instanciée du programme de départ, c'est-à-dire que la nouvelle forme obtenue du programme ne contient pas de variable. Cette réduction se déroule comme suit : étant donné un programme arbitraire, P et un ensemble d'atomes M, la réduction de P par rapport à M (P^M) est le programme défini obtenu en :

1. Supprimant toutes les règles qui ont un littéral *not* dans leur corps et dont le dit littéral se trouve dans M ;
2. Supprimant tous les littéraux *not* dans le corps des règles restantes.

Après cette réduction, un modèle M du programme défini P^M est le plus petit ensemble *close* (ne contenant pas de variables libres) par rapport à P^M. Cela correspond à un *modèle de Herbrand minimal*, qui est également un *point fixe* de l'*opérateur de conséquence* T_P.

$$T_P : 2^{\mathbb{X}} \rightarrow 2^{\mathbb{X}}$$
$$A \longmapsto T_P(A) = \{a | a \leftarrow b_1, \ldots, b_n \in P \wedge b_i \in A, i = 1, \ldots, n\}$$

$\mathbb{X}$ est l'ensemble d'atome de P.

2.2 Théorie de Dempster Shafer

La théorie de Dempster Shafer, également connue sous le nom de théorie des preuves ou théorie des fonctions de croyance, est un outil généralisant le raisonnement dans un contexte d'incertitude.

2.2.1 Concepts de base

Cette théorie est élaborée autour de la notion de *fonction de masse* qui, étant donnée une question, associe un poids lié aux évidences disponibles sur les différentes hypothèses associées [23] [36] [17] [26]. Elle se formalise de la manière suivante :
Soit $\Theta = \{\theta_1, \ldots, \theta_n\}$ un ensemble d'hypothèses de réponse à une question, cet ensemble est aussi appelé *cadre de discernement* (*frame of discernment, FOD*).
Soit 2^Θ l'ensemble des partitions de Θ, $2^\Theta = \{A | A \subseteq \Theta\}$.
Une fonction de masse m permettant la distribution de masses de probabilité sur l'ensemble des partitions est :

$$\begin{aligned} m : 2^\Theta &\to [0,1] \\ A &\longmapsto m(A) \\ \sum \{m(A)/A \subseteq \Theta\} &= 1. \end{aligned}$$

$m(\emptyset) = 0$ est la forme normalisée de la fonction de distribution et correspond à l'hypothèse du monde clos.
Si $\forall A \subseteq \Theta$ si $m(A) > 0$ alors A est un *ensemble focal* (*focal set*) pour m.
À partir de la distribution, $m(A)$, il est possible de déduire :

- La *fonction de croyance* $Bel()$:
  $\begin{cases} Bel(A) = \sum_{B \subseteq A} m(B) \\ m(\emptyset) = 0 \end{cases}$
  $Bel(A)$ est interprété comme le degré de croyance que la vérité réside en A.
- La fonction de plausibilité $Pl()$ d'une hypothèse A est la quantité de croyance non strictement engagée dans le complément de $\bar{A}$
  $\begin{cases} Pl : 2^\Theta \to [0,1] \\ Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) = 1 - Bel(\overline{\mathrm{A}}), A \subseteq \Theta \end{cases}$

3 Approche d'hybridation ASP-DST

La méthodologie que nous proposons comporte deux parties. Dans un premier temps, elle combine le programme ASP et les fonctions de croyance, en définissant une distribution de masse d'évidence sur l'ensemble des atomes de la base de Herbrand du programme. Cette distribution représente le niveau de véracité attribué aux atomes de la base de Herbrand et permettra ainsi d'évaluer la croyance d'un modèle.
Par la suite, nous utilisons la croyance de l'ensemble réponse et l'incohérence induite par les connaissances utilisées, pour calculer une valeur de décision en se basant sur une moyenne pondérée ordonnée.

3.1 Distribution d'évidence sur la base de Herbrand

Soit P un programme ASP défini, et B_P la base de Herbrand obtenue à partir de P.
B_P sera considérée comme le *FOD* de la fonction de distribution de masse, car tout modèle de P est un modèle de Herbrand minimal, qui est sous-ensemble de B_P.
Soit $2^{B_P} = \{A | A \subseteq B_P\}$ l'ensemble des partitions de B_P.
Soit m la fonction de distribution de masse d'évidence sur l'ensemble des partitions de la base de Herbrand.

$$\begin{aligned} m : 2^{B_P} &\to [0,1] \\ A &\longmapsto m(A) \\ \sum \{m(A)/A \subseteq B_P\} &= 1. \end{aligned}$$

De cette fonction de distribution, nous pouvons évaluer la croyance d'un sous-ensemble A de B_P ensemble solution d'un programme P : $Bel(A) = \sum_{B \subseteq A} m(B)$. Cela permet par conséquent de pouvoir évaluer la croyance d'un modèle du programme P.
Étant donné qu'il est possible d'avoir plusieurs ensembles modèles pour un programme ASP normal donné, nous ne nous intéressons plus au modèle minimal, mais plutôt au modèle ayant la croyance la plus élevée.

3.2 Inconsistance des programmes ASP

Les bases de connaissances construites à la main ou par l'automatisation sont souvent inexactes, ce qui signifie qu'elles ne reflètent pas, dans une certaine mesure, la réalité, et il est important de savoir à quel point elles s'écartent de celle-ci, car leur utilisation peut conduire à de mauvaises décisions [35]. En effet, cette mauvaise qualité des bases de connaissances est souvent due à des données erronées, à des informations manquantes ou à l'incohérence des schémas [10]. Cependant, malgré les approches utilisées pour les construire, il est encore difficile d'éviter ces problèmes lors de la formalisation des bases de connaissances et cela peut entraîner l'existence d'éléments d'informations contradictoires dans celle-ci [31].Ce problème de la représentation de la connaissance est généralisé sous le nom d'inconsistance.
En général, l'inconsistance peut se définir comme la présence de contradictions dans une base de connaissances, par rapport au formalisme de représentation [6]. Pour ce qui est de la représentation basée sur le formalisme du langage ASP, elle peut se traduire par tous les ensembles solutions sont inconsistant [33].
À fin de pouvoir quantifier ces problèmes rencontrés dans les bases de connaissances, différentes fonctions de mesure d'inconsistance ont été élaborées pour certaines méthodes de représentation de connaissances, comme la logique[30] et leur définition a été étendue à la programmation par ensemble réponses. À cet effet, la mesure de l'incohérence peut être exprimée par une fonction I positive, avec la sémantique de représenter la gravité de l'inconsistance dans

la base de connaissances, telle que I augmente avec l'incohérence.
Pour le cas de la représentation des connaissances basée sur ASP, [33] a défini des fonctions de mesure d'incohérence qui supportent le postulat de rationalité sans être *monotone*. Cette inconsistance est définie comme suite :
$I_{\pm}(P) = min\{|A| + |D|$ où A, D représentent respectivement les règles qui ont été ajoutées et réduites du programme P tel que $(P \cup A) - D$ est consistant $\}$.
La mesure de l'inconsistance des bases de connaissances construite à partir de la logique est généralement d'une complexité de Co-NP ou NP. Mais certaines fonctions permettent de réduire cette complexité, comme par l'exemple l'utilisation d'une mesure portée sur les ensembles réponses ou l'emploie de valeurs booléennes :

— $I_{\#}(P)$ = $min\{k|M$ est $k - inconsistant\}$, M est un ensemble solution du programme P et un ensemble solution est k-inconsistant, si il existe exactement k atomes et leurs opposés dans l'ensemble solution ;
— la mesure drastique [32]
$I_d(K) = \begin{cases} 1 \rightarrow inconsistance \\ 0 \end{cases}$
qui retourne 1 si le programme est inconsistant et 0 dans le cas contraire.

3.3 Moyenne pondérée ordonnée de la programmation des ensembles de réponses

Cette section exprime un moyen de calculer une valeur unique sur laquelle on peut prendre une décision à partir d'une représentation de la connaissance d'un problème avec un programme ASP.
Soit P un programme ASP normal. Soit X un ensemble solution du programme P et $Bel(X)$ sa croyance (voir section 3.1).
Soit $I(P) \in [0, 1]$ l'incohérence normalisée du programme P. Étant donné ces valeurs, un moyen approprié de combiner les métriques (croyance et incohérence) est l'utilisation d'une fonction d'agrégation, par exemple à l'aide d'une moyenne pondérée ordonnée (**OWA**) [9].
Cet opérateur peut être étendu en **GOWA**, qui est une forme généralisée de **OWA** [16]. Nous exploitons cet opérateur en faisant usage de la transformation qui range les valeurs de l'inconsistance et de la croyance dans l'ordre croissant :
$GOWA_{\alpha,\lambda}(I(P), Bel(X)) = (\alpha_1 * max^{\lambda}(I(P), Bel(X)) + \alpha_2 * min^{\lambda}(I(P), Bel(X)))^{\frac{1}{\lambda}}$
avec $w = (w_1, w_2) \in [0, 1]^2$, et $w_1 + w_2 = 1$:
w_1, w_2 représentent la confiance que l'on a respectivement sur la formalisation de la représentation de la connaissance et l'ensemble solution émanant de cette représentation. La figure 1 ci-dessous résume notre approche mettant en contribution la programmation par l'ensemble et la théorie de Dempster Shafer.

Example d'illustation
Soit P un programme ASP avec les règles suivantes :
$oiseau(X) : -vole(X), plume(X), not\ anormal(X).$
$\{vole(X); nage(X)\} : -oiseau(X).$

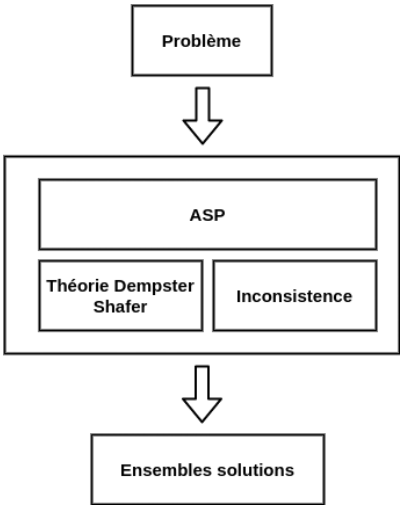

FIGURE 1 – Approche DST-ASP

$oiseau(tweety).$

— Pour améliorer cette base de connaissances (retirer les inconsistances) nous allons :
Ajouter la contrainte selon laquelle un oiseau ne peut pas avoir les aptitudes de nager et voler simultanément. La base de connaissances devient :
$oiseau(X) : -vole(X), plume(X), not\ anormal(X).$
$\{vole(X); nage(X)\} : -oiseau(X).$
$: -nage(X), vole(X).$
$oiseau(tweety).$
— La base de Herbrand associée à ce programme est :
$B_P = \{vole(tweety), oiseau(tweety)$
$anormal(tweety), plume(tweety), nage(tweety)\}.$

— Pour réunir *ASP* et *DST*, une assignation de croyance de base (BBA) doit être définie sur 2^{H_P}. Comme cette assignation peut devenir difficile, sur tout avec une taille importante de H_P (complexité NP), nous allons faire usage d'une fonction de raffinement.
De ce fait, il faut :

1. Utiliser les connaissances ou l'expérience passées pour regrouper la base Herbrand en ensembles disjoints.
2. Attribuer une valeur de preuve aux éléments individuels de la base Herbrand.
3. Définir une affectation de croyance sur le nouvel ensemble qui constitue une partition de la base Herbrand.

Les croyances que nous obtenons par rapport à la distribution 1 sont :

1. $X1 = \{oiseau(tweety)\}$ avec une croyance de :
$Bel(X1) = 0.2$
2. $X2 = \{oiseau(tweety), vole(tweety)\}$
$Bel(X2) = m(\{oiseau(tweety)\}) +$

{ vole(tweety) } : 0.1	{ oiseau(tweety)} : 0.2
{ nage(tweety)} : 0.1	{ plume(tweety)} : 0.1
{ vole(tweety) , nage(tweety)} :0.1	{ vole(tweety) , oiseau(tweety)} : 0.2
{ vole(tweety) , plume(tweety)} :0.1	{nage(tweety) , oiseau(tweety)} : 0.1

TABLE 1 – Distribution de masses

$m(\{vole(tweety)\}) + m(\{oiseau(tweety), vole(tweety)\})$
$Bel(X2) = 0.2 + 0.1 + 0.2 = 0.5$

3. $X3 = \{oiseau(tweety), nage(tweety)\}$
$Bel(X3) = m(\{oiseau(tweety)\}) + m(\{nage(tweety)\}) + m(\{oiseau(tweety), nage(tweety)\})$
$Bel(X3) = 0.2 + 0.1 + 0.1 = 0.4$

— Notre représentation de connaissance a une inconsistance de :
$I_{\pm} = 1$ Nous normalisons cette valeur par $\frac{I_{\pm}}{|A|+|P|}$ où A représente les règles ajoutées et P le programme initial sans ajout ni réduction.
$I = 0.25$

— De ce qui précède, nous pouvons calculer les valeurs de décision pour les deux modèles $X1$, $X2$ et $X3$: En utilisant GOWA

1. $X1 = \{oiseau(tweety)\}$ La croyance de $X1$ par rapport à la distribution m est :
$Bel(X1) = 0.2$
$w_1 = 0.5, w_2 = 0.5$
$GOWA_{w,1}(P, X) = 0.5 * 0.25 + 0.5 * 0.2 = 0.22$
$I1 = 0.22$
2. $X2 = \{oiseau(tweety), vole(tweety)\}$
La croyance de $X2$ par rapport à la distribution m est :
$Bel(X2) = 0.5$
$w_1 = 0.5, w_2 = 0.5$
$GOWA_{w,1}(P, X) = 0.5 * 0.25 + 0.5 * 0.5 = 0.37$
$I2 = 0.37$
3. $X3 = \{oiseau(tweety), nage(tweety)\}$
La croyance de $X3$ par rapport à la distribution m est :
$Bel(X3) = 0.4$
$w_1 = 0.5, w_2 = 0.5$
$GOWA_{w,1}(P, X) = 0.5 * 0.25 + 0.5 * 0.4 = 0.32$
$I3 = 0.32$

L'exemple nous montre qu'il serait judicieux de choisir la solution $X2$ ou $X3$, au lieu de $X1$, car elles ont des valeurs de décision supérieures à celle de $X1$. Cette différence de valeurs peut s'expliquer par le fait que la croyance du modèle $X1$ est faible comparée à celle des modèles $X3$ et $X2$.

De cette example la solution $X2 = \{oiseau(tweety), vole(tweety)\}$ est meilleure vu qu'elle a la plus grande valeur de décision.

4 Comparaison et Discussion

4.1 Comparaison

Nous allons dans un prémier temps comparer notre approche hybride avec les deux modèles mise en contribution de façon individuelle et par la suite avec une la proposition de Al Machot et al [2].
En effet, pour un même problème entre l'approche hybride et l'approche ASP toute seule, la différence fondamentale est que cette dernière ne donne aucun moyen de choisir une solution parmi l'ensemble des modèles trouvés pour le problème, ce qui peut être embarrassant pour la prise de décision. Par ailleurs pour la même condition précédente, l'approche Dempter Shaffer toute seule pourra à la limite fournir des valeurs de croyance, sans qu'on ne sache quelles sont les solutions du problème. Le tableau 2 ci-dessous récapitule la comparaison entre l'approche hybride et les deux autres approches prises seules.
En résumé, l'approche hybride que nous proposons permet de combler les limites des méthodes ASP et DSP. Par conséquent, elle permet de résoudre un problème et sélectionner la meilleure des solutions.

Méthode	Solution au problème	Choix d'une solution
ASP	oui	non
DST	non	oui
Hybridation	oui	oui

TABLE 2 – Comparaison ASP, DST et Hybridation

Pour ce qui est de la comparaison avec l'approche élaborée par Al Machot et al. [2], il en ressort que :

— Nous proposons une généralisation de la combinaison entre la programmation par ensemble et la théorie de Dempster shafer alors que les travaux de Al Machot et al. présentent une façon et un cas particulier de cette intégration et s'intéressent à la combinaison des sources différentes.
— Dans l'approche que nous présentons, nous tenons compte de l'inconsistance, qui peut considérablement influencer la prise de décision comparé à la seule croyance. Par contre cet aspect de la connaissance n'est pas considéré par ces auteurs.

4.2 Discussion

Nous proposons une hybridation de programmation par l'ensemble (ASP) et de la théorie de Demspter Shafer qui permet de sélectionner la meilleure solution au cours d'un processus d'expertise. En effet ce processus permet d'explorer toutes les pistes de solutions possibles d'un problème et de ce fait il devient import d'avoir une value de décision pour faciliter le choix d'une possibilité par rapport aux autres. Cette approche se résume aux étapes suivantes :

— Dans un premier temps de pouvoir évaluer la croyance liée à un ensemble solution que représente les différentes possibilités d'un processus d'expertise.
— Par la suite, nous calculons l'incohérence de la base de connaissances ou des ensembles solutions.
— En fin les deux valeurs précédemment calculées nous permettent d'avoir une valeur de décision, qui permet de choisir un ensemble solution en tenant compte de l'inconsistance et la croyance.

Pour l'implémentation, nous avons développé un prototype en langage Python, qui utilise les librairies py_dempster_shafer [1] et clyngor [2]. Techniquement nous utilisons une structure de données de type dictionnaire pour associer la distribution de masse aux éléments de la base de Herbrand et par la suite ce dictionnaire est exploité pour le calcul de croyance et la valeur de décision.
En raison de la complexité que l'on peut avoir lorsque la base de Herbrand est important ou qu'il devient difficile de définir une fonction de distribution d'évidence, une approche qui reste dans la même démarche, que la précédente consistera à utiliser la fonction de support simplifiée, qui est une fonction de croyance basée sur le support d'un sous-ensemble [25] de la dite base.
En effet elle consiste à fixer un support $s \in [0, 1]$ représentant la croyance pour un sous-ensemble $A \subset H_P$ et tout sous-ensemble de H_P le contenant. Elle se décrit comme suit

$$S(B) = \begin{cases} 0, A \nsubseteq B \\ s, A \subseteq B \neq H_P \\ 1, B = H_P \end{cases}$$

Pour illustrer, utilisons l'exemple précédent et supposons que la personne accorde un support de $s = 0.6$ au sous ensemble $\{oiseau(tweety), vole(tweety)\}$. Cela revient aux calculs de croyance suivants :

— $X1 = \{oiseau(tweety)\}$
$X1 \subseteq \{oiseau(tweety), vole(tweety)\}$
$Bel(X1) = 0.6$
— $X2 = \{oiseau(tweety), vole(tweety)\}$
$X2 \subseteq \{oiseau(tweety), vole(tweety)\}$
$Bel(X2) = 0.6$
— $X3 = \{oiseau(tweety), nage(tweety)\}$
$X3 \nsubseteq \{oiseau(tweety), vole(tweety)\}$
$Bel(X3) = 0$

5 Conclusion

Dans ce travail, nous avons dans un premier temps présenté la programmation par ensemble réponses qui est activement utilisée dans la représentation des connaissances et pour le raisonnement non-monotone. Par la suite, nous avons présenté une synthèse de la théorie de Dempster Shafer utilisée pour représenter des connaissances incertaines. Enfin, nous avons défini une méthodologie de calcul d'une valeur décisionnelle, qui premièrement combine les deux théories précédentes pour un meilleur choix du modèle en fonction de la croyance. Cette croyance est par la suite combinée à la valeur d'incohérence de la base de connaissances au travers de la moyenne pondérée ordonnée généralisée pour avoir une valeur de prise de décision.
L'approche proposée dans ce travail permet de prendre une décision non seulement en se focalisant sur l'ensemble solution de la programmation par ensemble, mais en tenant compte de la croyance et l'incohérence.
Pour ce qui est des difficultés liées à la complexité, nous proposons l'utilisation d'un simple fonction d'assignation basée sur un support d'évidence pou la difficulté de distribution d'évidence et l'utilisation d'inconsistance drastique pour la complexité liée à la mesure d'inconsistance.
En effet, cette approche facilite la prise de décision, car elle est numérique et de plus permet de prendre une meilleure décision parce qu'elle tient compte de la croyance et l'incohérence émanant du modèle de représentation du problème.
En comparaison avec l'approche proposée par Al Machot et al [2] montre que notre méthode est général et tient compte de l'inconsistance, ce qui n'est pas le cas des travaux de ces auteurs. De plus notre méthode offre un mécanisme de décision sur les solutions d'un problème, que ne dispose pas ASP et DST prisent individuellement.
Pour la suite de ce travail, nous allons dans un premier temps définir un algorithme qui permettra de trouver le modèle avec la plus grande conviction, basée sur la croyance et l'incohérence. Par la suite développer un outil basé sur cet algorithme et qui intègre les deux librairies du langage Python que nous avons utilisé.

1. https ://github.com/reineking/pyds
2. https ://github.com/aluriak/clyngor

Références

[1] Erdi Aker, Volkan Patoglu, and Esra Erdem. Answer set programming for reasoning with semantic knowledge in collaborative housekeeping robotics. *IFAC Proceedings Volumes*, 45(22) :77–83, 2012.

[2] Fadi Al Machot, Heinrich C Mayr, and Suneth Ranasinghe. A hybrid reasoning approach for activity recognition based on answer set programming and dempster–shafer theory. In *Recent Advances in Nonlinear Dynamics and Synchronization*, pages 303–318. Springer, 2018.

[3] Chitta Baral. Logic programming and uncertainty. In *International Conference on Scalable Uncertainty Management*, pages 22–37. Springer, 2011.

[4] Kim Bauters, Steven Schockaert, Martine De Cock, and Dirk Vermeir. Semantics for possibilistic answer set programs : uncertain rules versus rules with uncertain conclusions. *International journal of approximate reasoning*, 55(2) :739–761, 2014.

[5] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12) :92–103, 2011.

[6] Mark Burgin and CNJ de Vey Mestdagh. Consistent structuring of inconsistent knowledge. *Journal of Intelligent Information Systems*, 45(1) :5–28, 2015.

[7] Federico Cerutti and Matthias Thimm. A general approach to reasoning with probabilities. *International Journal of Approximate Reasoning*, 111 :35–50, 2019.

[8] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Answer set programming : A primer. In *Reasoning Web International Summer School*, pages 40–110. Springer, 2009.

[9] Ali Emrouznejad and Marianna Marra. Ordered weighted averaging operators 1988–2014 : A citation-based literature survey. *International Journal of Intelligent Systems*, 29(11) :994–1014, 2014.

[10] Luis Galárraga, Simon Razniewski, Antoine Amarilli, and Fabian M Suchanek. Predicting completeness in knowledge bases. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 375–383, 2017.

[11] Laurent Garcia, Claire Lefèvre, Odile Papini, Igor Stephan, and Eric Würbel. Possibilistic asp base revision by certain input. 2018.

[12] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice. *Synthesis lectures on artificial intelligence and machine learning*, 6(3) :1–238, 2012.

[13] Jeroen Janssen, Steven Schockaert, Dirk Vermeir, and Martine De Cock. *Answer Set Programming for Continuous Domains : A Fuzzy Logic Approach*, volume 5. Springer Science & Business Media, 2012.

[14] Antonis C Kakas. Default reasoning via negation as failure. In *Foundations of Knowledge Representation and Reasoning*, pages 160–178. Springer, 1994.

[15] Michael Kaminski. A note on the stable model semantics for logic programs. *Artificial intelligence*, 96(2) :467–479, 1997.

[16] Fateme Kouchakinezhad and Alexandra Šipošová. Ordered weighted averaging operators and their generalizations with applications in decision making. *Iranian Journal of Operations Research*, 8(2) :48–57, 2017.

[17] Liping Liu and Ronald R Yager. Classic works of the dempster-shafer theory of belief functions : An introduction. In *Classic works of the Dempster-Shafer theory of belief functions*, pages 1–34. Springer, 2008.

[18] Peter Lucas and Linda Van Der Gaag. *Principles of expert systems*. Addison-Wesley Wokingham, 1991.

[19] Huver Loisel Peyrouty Pineau Tuffery M. Peyrouty, Chanay ; Fourniguet. Recommandations pour l'application de la norme nf x 50-110 :2003. Technical report, Association Française de Normalisation, 2011.

[20] Pierre Marquis, Odile Papini, and Henri Prade. Panorama de l'intelligence artificielle. *Ses Bases Méthodologiques, ses Développements*, 2, 2014.

[21] Arnaud Martin, Anne-Laure Jousselme, and Christophe Osswald. Conflict measure for the discounting operation on belief functions. In *2008 11th International conference on information fusion*, pages 1–8. IEEE, 2008.

[22] Steve Pye, Francis GN Li, Arthur Petersen, Oliver Broad, Will McDowall, James Price, and Will Usher. Assessing qualitative and quantitative dimensions of uncertainty in energy modelling for policy support in the united kingdom. *Energy research & social science*, 46 :332–344, 2018.

[23] Thomas Reineking. *Belief functions : theory and algorithms*. PhD thesis, Universität Bremen, 2014.

[24] Fabrizio Riguzzi. *Foundations of Probabilistic Logic Programming*. River Publishers, 2018.

[25] Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

[26] Glenn Shafer. Probability judgment in artificial intelligence. In *Machine Intelligence and Pattern Recognition*, volume 4, pages 127–135. Elsevier, 1986.

[27] Yi-Dong Shen and Thomas Eiter. Determining inference semantics for disjunctive logic programs (extended abstract).

[28] Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2) :181–234, 2002.

[29] Philippe Smets. Imperfect information : Imprecision and uncertainty. In *Uncertainty management in information systems*, pages 225–254. Springer, 1997.

[30] Matthias Thimm. On the expressivity of inconsistency measures. *Artificial Intelligence*, 234 :120–151, 2016.

[31] Matthias Thimm. Inconsistency measurement. In *International Conference on Scalable Uncertainty Management*, pages 9–23. Springer, 2019.

[32] Matthias Thimm and Johannes P Wallner. On the complexity of inconsistency measurement. *Artificial Intelligence*, 275 :411–456, 2019.

[33] Markus Ulbricht, Matthias Thimm, and Gerhard Brewka. Measuring inconsistency in answer set programs. In *European Conference on Logics in Artificial Intelligence*, pages 577–583. Springer, 2016.

[34] Yi Wang and Joohyung Lee. Handling uncertainty in answer set programming. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[35] Michael Wick, Sameer Singh, Ari Kobren, and Andrew McCallum. Assessing confidence of knowledge base content with an experimental study in entity resolution. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 13–18, 2013.

[36] Ronald R Yager and Liping Liu. *Classic works of the Dempster-Shafer theory of belief functions*, volume 219. Springer, 2008.