HAL
archives-ouvertes.fr

Une approche de l'adaptation en raisonnement à partir de cas fondée sur l'optimisation sous contraintes

Julien Cojan, Jean Lieber

▶ **To cite this version:**

Julien Cojan, Jean Lieber. Une approche de l'adaptation en raisonnement à partir de cas fondée sur l'optimisation sous contraintes. Journées d'Intelligence Artificielle Fondamentale, Oct 2009, Marseille, France. inria-00425030

HAL Id: inria-00425030
https://hal.inria.fr/inria-00425030

Submitted on 19 Oct 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Une approche de l'adaptation en raisonnement à partir de cas fondée sur l'optimisation sous contraintes

Julien Cojan, Jean Lieber

UHP-Nancy1, LORIA,
BP 239, 54506 Vandœuvre-lès-Nancy

Résumé : Une approche de l'adaptation en raisonnement à partir de cas (RÀPC) qui repose sur la théorie du changement minimal est présentée ici dans un formalisme plus général que la logique propositionnelle finie. Ce formalisme permet d'exprimer des valeurs numériques ce qui est utile dans le cadre du RÀPC. Cette approche est réduite à un problème d'optimisation, et sous hypothèses, à un problème de programmation linéaire.

Mots-clés : raisonnement à partir de cas, adaptation, théorie du changement minimal, optimisation, programmation linéaire.

1 Introduction

Le raisonnement à partir de cas (RÀPC, Riesbeck & Schank (1989)) est un modèle de résolution de problèmes à partir d'expérience. Cette expérience est constituée d'un ensemble d'épisodes de résolution de problèmes : les cas sources.

On s'intéresse dans cet article à la phase d'adaptation qui consiste à ajuster la solution d'un cas source au nouveau problème. Le RÀPC est à l'origine une théorie en sciences cognitives pour laquelle il n'y a pas de théorie formelle logique qui fasse consensus, en particulier pour l'adaptation. Une formalisation de cette phase du RÀPC a été proposée dans Lieber (2007) dans le cadre de la logique propositionnelle. Elle repose sur la théorie de la révision des croyances qui consiste à agréger deux bases de connaissance potentiellement contradictoires en abandonnant, si nécessaire, des informations de l'une des deux bases afin de préserver la cohérence du résultat. Dans Lieber (2007), l'adaptation de la solution d'un cas source est obtenue en la révisant par le contexte du nouveau problème afin qu'elle s'y applique. Une approche de la révision est étudiée ici sous l'angle de l'optimisation afin d'étendre cette approche de l'adaptation à un formalisme permettant d'exprimer des contraintes numériques (réelles ou entières).

La section 2 est consacrée à l'adaptation en RÀPC et à une approche d'adaptation fondée sur un opérateur de révision. Dans la section 3, après un rappel sur la théorie de la révision des connaissances en logique propositionnelle, une extension de cette théorie à un formalisme plus général est proposée. On étudiera le cas particulier d'un opérateur de révision défini à partir d'une distance et de connaissances représentées par

des simplexes en section 4. Cette section constitue la contribution la plus importante de cet article. Elle montre comment une certaine classe de problèmes de révision peut se ramener à de la programmation linéaire mixte et comment cela s'applique à un exemple d'adaptation de recette de cuisine. La section 5 conclut en décrivant l'état actuel de cette étude.

2 L'adaptation en RÀPC

Le raisonnement à partir de cas (RÀPC) est un paradigme de résolution de problèmes à l'aide de l'expérience de la résolution d'autres problèmes appelés cas sources et regroupés dans une base finie de cas. La distinction *a priori* entre problème et solution n'étant pas toujours pertinente (cf. l'exemple en section 4.3), nous ne la ferons pas ici.

Le but d'un système de RÀPC est de préciser un cas `Cible`, pour lequel il manque des informations, en `CibleComplété`. Cela se fait classiquement en deux étapes :

- La *remémoration* d'un cas `Srce` dans la base de cas, jugé pertinent par rapport à `Cible`, par exemple par leur similarité. On ne s'intéresse pas à cette étape ici, dans la suite `Srce` sera supposé déjà connu.
- L'*adaptation* qui permet de préciser `Cible` en `CibleComplété`, sur la base de `Srce`.

Si une partie problème et une partie solution peuvent être identifiées, `Cible` est constitué d'une partie problème spécifiée et d'une partie solution à spécifier. L'adaptation consiste alors à modifier la partie solution de `Srce` pour qu'elle réponde au problème de `Cible`. D'autres connaissances peuvent accompagner la base de cas, en particulier les connaissances du domaine (`CD`) qui apportent des connaissances complémentaires sur le domaine de l'application.

2.1 Exemple en logique propositionnelle

Cet exemple est motivé par l'application Taaable (`http://www.taaable.fr`, Badra *et al.* (2009)). Il s'agit d'un système de proposition de recettes de cuisine s'appuyant sur une base de recettes et fonctionnant suivant le principe du RÀPC. Pour simplifier, on ne considère que la liste des ingrédients des recettes. Les requêtes et les réponses (recettes) sont des formules sur des variables propositionnelles représentant les ingrédients. Supposons qu'à partir d'une recette de salade de tomates on veuille faire une recette avec les ingrédients dont on dispose, à savoir des concombres, des poivrons de la feta et de la vinaigrette.

$$\begin{aligned} \texttt{Srce} &= \text{tomate} \wedge \text{feta} \wedge \text{vinaigrette} \wedge \neg\text{poivron} \wedge \neg\text{concombre} \\ \texttt{Cible} &= \neg\text{tomate} \wedge \neg\text{poivron} \\ \texttt{CD} &= \text{légume-fruit} \Leftrightarrow \text{tomate} \vee \text{concombre} \vee \text{poivron} \end{aligned}$$

La restriction aux ingrédients disponible s'exprime dans `Cible` par l'interdiction des autres ingrédients, ici les tomates et les poivrons. `CD` exprime le fait que parmi les ingrédients considérés, les légumes-fruits sont les tomates, les concombres et les poivrons.

2.2 $\dot{+}$-adaptation

Lieber (2007) propose une formalisation de l'adaptation par la théorie de la révision de connaissances. Cette théorie traite de l'évolution d'une base de connaissances pour

être assimilée à d'autres connaissances jugées plus fiables. Cela est formalisé par un opérateur $\dot{+}$ qui à deux bases de connaissances ψ et μ associe la révision de ψ par μ : $\psi \dot{+} \mu$. La $\dot{+}$-adaptation présentée dans Lieber (2007) consiste à considérer les cas comme des connaissances dans un contexte (CD) et à formaliser l'adaptation, i.e. les modifications à apporter aux connaissances de Srce : (CD $\wedge$ Srce) pour le rendre applicable dans le contexte de Cible : (CD $\wedge$ Cible) par un opérateur de révision $\dot{+}$:

$$\texttt{CibleComplété} = (\texttt{CD} \wedge \texttt{Srce}) \dot{+} (\texttt{CD} \wedge \texttt{Cible})$$

Dans l'exemple de la section 2.1, avec l'opérateur de révision de Dalal (cf. ex. 1) :

$$\texttt{CibleComplété} = \text{concombre} \wedge \text{feta} \wedge \text{vinaigrette} \wedge \neg\text{tomate}$$

La section suivant est consacrée à l'étude d'un opérateur de révision dans un formalisme approprié au type de problème de RÀPC présenté en section 4.3.

3 Révision en logique propositionnelle et extension

3.1 Rappel : la révision en logique propositionnelle

La révision est un aspect de la théorie du changement minimal. Étant donné deux bases de connaissances ψ et μ où μ est jugée plus fiable que ψ, la révision de ψ par μ consiste à incorporer ψ à μ en préservant, si possible, la cohérence du résultat, quitte à sacrifier certaines parties de ψ. La théorie AGM (Alchourrón *et al.* (1985)) donne une caractérisation logique de la révision sous la forme d'un ensemble de postulats qui établit les propriétés attendues d'un opérateur $\dot{+}$ où $\psi \dot{+} \mu$ est le résultat de la révision de ψ par μ. Katsuno & Mendelzon (1991) donnent une formulation de ces postulats en logique propositionnelle finie. Étant donné des formules ψ, ψ_2, μ, μ_2 et ϕ :

- $(R1)$ $\psi \dot{+} \mu \vDash \mu$
- $(R2)$ Si $\psi \wedge \mu$ est satisfaisable alors $\psi \dot{+} \mu \equiv \psi \wedge \mu$.
- $(R3)$ Si μ est satisfaisable alors $\psi \dot{+} \mu$ l'est aussi.
- $(R4)$ Si $\psi \equiv \psi_2$ et $\mu \equiv \mu_2$ alors $\psi \dot{+} \mu \equiv \psi_2 \dot{+} \mu_2$.
- $(R5)$ $(\psi \dot{+} \mu) \wedge \phi \vDash \psi \dot{+} (\mu \wedge \phi)$
- $(R6)$ Si $(\psi \dot{+} \mu) \wedge \phi$ est satisfaisable alors $\psi \dot{+} (\mu \wedge \phi) \vDash (\psi \dot{+} \mu) \wedge \phi$.

Ces postulats caractérisent $\dot{+}$ au niveau logique, Katsuno et Mendelzon proposent aussi une caractérisation équivalente au niveau des interprétations à partir de la notion d'assignation fidèle. Une *assignation fidèle* est une fonction qui associe à chaque formule ψ un préordre $\leqslant_\psi$ sur les interprétations tel que :

- $(A1)$ Si $I, I' \in$ Modèles(ψ) alors $I \not<_\psi I'$.
- $(A2)$ Si $I \in$ Modèles(ψ) et $I' \notin$ Modèles(ψ) alors $I <_\psi I'$.
- $(A3)$ Si $\psi \equiv \phi$ alors $\leqslant_\psi = \leqslant_\phi$.

où $I <_\psi I'$ ssi ($I \leqslant_\psi I'$ et $I' \not\leqslant_\psi I$) et Modèles(ψ) est l'ensemble des modèles de ψ.

Théorème 1 (Katsuno & Mendelzon (1991))
Un opérateur de révision $\dot{+}$ satisfait les postulats $(R1)$ à $(R6)$ si et seulement si il existe une assignation fidèle telle que :

$$\textit{Modèles}(\psi \dot{+} \mu) = \textit{Min}(\textit{Modèles}(\mu), \leqslant_\psi)$$

où $\textit{Min}(A, \leqslant) = \{x \in A \mid \forall y \in A,\ y \not< x\}$.

Exemple 1 (révision de Dalal)
L'opérateur de révision de Dalal est défini à partir de la distance de Hamming d entre interprétations en posant l'assignation fidèle suivante :

$$I \leqslant_\psi I' \quad \textit{ssi} \quad d(\textit{Modèles}(\psi), I) \leqslant d(\textit{Modèles}(\psi), I')$$

Notons que si on substitue d par une autre distance entre interprétations, l'opérateur obtenu, satisfera également $(R1)$ à $(R6)$.

Pour une application dans le cadre du RÀPC, nous nous intéressons dans la suite à une extension à un formalisme permettant la représentation de valeurs réelles et entières.

3.2 Extension : de la logique propositionnelle à un cadre plus large

Soit $(\mathcal{L},\mathcal{U})$ un formalisme de représentation avec $\mathcal{L}$ un ensemble de formules et $\mathcal{U}$ un ensemble dont les éléments sont appelés interprétations. Soit Modèles : $\psi \in \mathcal{L} \mapsto$ Modèles$(\psi) \in 2^{\mathcal{U}}$. Pour $x \in$ Modèles(ψ), x est appelé *modèle* de ψ. On suppose que si $f, g \in \mathcal{L}$, alors $f \wedge g \in \mathcal{L}$ et Modèles$(f \wedge g) =$ Modèles$(f) \cap$ Modèles(g). Par ailleurs on note $f \models g$ si Modèles$(f) \subseteq$ Modèles(g). Soit $\mathcal{P} = \{\text{Modèles}(\psi) \mid \psi \in \mathcal{L}\}$, $\mathcal{P}$ est l'ensemble des ensembles d'interprétations qui peuvent être représentés par le langage $\mathcal{L}$ et, par conséquent, $\mathcal{P}$ est clos pour $\cap$. Ce formalisme étend la logique propositionnelle finie : $\mathcal{U} = \{\texttt{faux}, \texttt{vrai}\}^n$ (où n est le nombre de variables) et $\mathcal{P} = 2^{\mathcal{U}}$ car pour tout ensemble d'interprétations $A \subseteq \mathcal{U} = \{\texttt{faux}, \texttt{vrai}\}^n$, il existe une formule φ telle que Modèles$(\varphi) = A$.

Comme précédemment (postulat $(R4)$), on adhère au principe de non pertinence de la syntaxe. Une formule ψ est donc assimilée à l'ensemble de ses modèles Modèles$(\psi) \in \mathcal{P}$. Le postulat $(R4)$ se traduit par la tautologie « si $A_1 = A_2$ et $B_1 = B_2$, alors $A_1 \dot{+} B_1 = A_2 \dot{+} B_2$ ». Les postulats $(R1)$ à $(R3)$ et $(R5)$, $(R6)$ se traduisent respectivement par les postulats suivants, pour $A, B, C \in \mathcal{P}$:

(R*év*1) $A \dot{+} B \subseteq B$
(R*év*2) Si $A \cap B \neq \emptyset$ alors $A \dot{+} B = A \cap B$.
(R*év*3) Si $B \neq \emptyset$ alors $A \dot{+} B \neq \emptyset$.
(R*év*4) $(A \dot{+} B) \cap C \subseteq A \dot{+} (B \cap C)$
(R*év*5) Si $(A \dot{+} B) \cap C \neq \emptyset$ alors $A \dot{+} (B \cap C) \subseteq (A \dot{+} B) \cap C$.

En logique propositionnelle ces postulats sont équivalents aux postulats initiaux.

Dans ce formalisme, une assignation fidèle est une fonction qui associe au éléments de $\mathcal{P}$ un préordre partiel sur $\mathcal{U}$. La définition dans le cadre de la logique propositionnelle

se transpose ici en substituant ψ et Modèles(ψ) par $A \in \mathcal{P}$ dans $(A1)$ et $(A2)$. $(A3)$ devient la tautologie « si $A = B$ alors $\leqslant_A = \leqslant_B$ ».

Le théorème 1 n'est plus valable ici. En effet, étant donné un préordre $\leqslant$ sur $\mathcal{U}$ et $A \subseteq \mathcal{U}$, rien ne garantit que $\text{Min}(A, \leqslant) \neq \emptyset$. Par exemple avec $\mathcal{U} = \mathbb{R}$, $\mathcal{P} = 2^{\mathcal{U}}$, $\leqslant$ l'ordre usuel sur $\mathbb{R}$ et $A =]1, 2]$, $\text{Min}(A, \leqslant) = \emptyset$. Il faut ajouter la condition suivante aux assignations fidèles :

$$(A4) \text{ pour } A, B \in \mathcal{P},\ B \neq \emptyset\ \ \text{Min}(B, \leqslant_A) \neq \emptyset$$

De plus la preuve de l'une des implications demande à ce que les sous ensembles finis de $\mathcal{U}$ soient dans $\mathcal{P}$, ce qui n'est pas nécessairement le cas ici.

Théorème 2 (Katsuno & Mendelzon (1991) adapté)

1. *Si $\leqslant_\bullet$ est une assignation fidèle satisfaisant $(A4)$, alors l'opérateur de révision $\dot{+}$ défini par Modèles$(A \dot{+} B) =$ Min$(B, \leqslant_A)$ satisfait les postulats* (Rév1) *à* (Rév5)
2. *Si $\mathcal{P}$ contient les sous-ensembles finis de $\mathcal{U}$, alors la réciproque est vraie.*

Schéma de preuve : La preuve du « si » (1.) de Katsuno & Mendelzon (1991) reste valide ici, Pour le postulat (*Rév*3) la propriété $(A4)$ est cependant implicite car toujours satisfaite en logique propositionnelle finie.

La preuve du « seulement si » (2.) de Katsuno & Mendelzon (1991) se sert de l'assignation fidèle $\leqslant_\bullet$ définie à partir de $\dot{+}$ par : $x \leqslant_A y$ ssi $x \in A \dot{+} \{x, y\}$, pour $A \in \mathcal{P}$ et $x, y \in \mathcal{U}$. Elle reste valide dans ce formalisme. Le postulat (*Rév*3) garantit que $\leqslant_\bullet$ satisfait bien la propriété $(A4)$. □

On se servira aussi dans la suite d'une généralisation de l'opérateur de Dalal défini à partir d'une distance d sur $\mathcal{U}$.

Définition 1

Pour d une distance sur $\mathcal{U}$, soit l'assignation fidèle $\leqslant^d_\bullet$ telle que pour $A \in \mathcal{P}$ et $x, y \in \mathcal{U}$:

$$x \leqslant^d_A y \quad \textit{ssi} \quad d(A, x) \leqslant d(A, y)$$

Pour que cette définition soit correcte (i.e. que $\leqslant^d_\bullet$ satisfasse $(A2)$) l'hypothèse de fermeture des éléments de $\mathcal{P}$ sous d est nécessaire : pour tout $A \in \mathcal{P}$ et $x \in \mathcal{U}$, $d(A, x) = 0$ doit entraîner que $x \in A$. De plus $\leqslant^d_\bullet$ peut ne pas satisfaire $(A4)$, c'est le cas avec la distance usuelle sur $\mathcal{U} = \mathbb{R}$ et $\mathcal{P} = 2^{\mathbb{R}}$, en prenant $A = \{0\}$ et $B =]1, 2]$ on obtient $\text{Min}(B, \leqslant^d_A) = \emptyset$. Des hypothèses supplémentaires sur $\mathcal{P}$ sont nécessaires pour assurer $(A4)$, par exemple que $\mathcal{P}$ soit inclus dans l'ensemble des compacts de $(\mathcal{U}, d)$.

Étant donné un opérateur de révision, le théorème de représentation précédant permet de réduire le calcul de la révision à un problème d'optimisation :

$$\text{Trouver les } x \text{ minimaux pour } \leqslant^d_A \text{ sous la contrainte } x \in B$$

En particulier, avec l'assignation fidèle $\leqslant^d_\bullet$, le problème d'optimisation devient :

$$\text{Trouver les } x \text{ minimisant } d(A, x) \text{ sous la contrainte } x \in B$$

Dans la suite nous nous intéressons aux conditions pour lesquelles la réduction à de la programmation linéaire est possible.

4 Application aux représentations à l'aide de simplexes

4.1 Rappel sur la programmation linéaire

Un problème de programmation linéaire consiste à déterminer les minima d'une forme linéaire sur un ensemble déterminé par une conjonction finie de contraintes linéaires. Plus précisément, soit $x_1, \ldots, x_n$, n variables, chaque variable x_i ayant pour domaine de définition un intervalle de $\mathbb{R}$ ou $\mathbb{Z}$. Les contraintes sont du type : $a_1x_1 + \ldots + a_nx_n \leqslant b$, et la fonction objectif à minimiser est du type $c_1x_1 + \ldots + c_nx_n$.

Dans la littérature de la recherche opérationnelle (voir, p.ex. Dantzig (1963)), le terme programmation linéaire concerne les problèmes pour lesquels toutes les variables prennent des valeurs réelles, c'est-à-dire que leur domaine de définition est un intervalle de $\mathbb{R}$. Dans ce cadre il existe des algorithmes de résolution polynômiaux par rapport au nombre de variables et de contraintes linéaires. Dans le cas où certaines variables prennent des valeurs entières, on parle de programmation linéaire mixte et la résolution devient NP-difficile.

4.2 Représentation à l'aide de simplexes

Pour se placer dans le cadre de la programmation linéaire (mixte), $\mathcal{U}$ est supposé être de la forme : $\mathcal{U} = I_1 \times \ldots \times I_n$ où chaque I_i est un intervalle de $\mathbb{R}$ ou $\mathbb{Z}$.

Une contrainte linéaire $\sum_i a_i x_i \leqslant b$ détermine un sous-espace de $\mathcal{U}$ que l'on note $[\sum_i a_i x_i \leqslant b] \in 2^{\mathcal{U}}$. $\mathcal{P}$ est l'ensemble des parties de $\mathcal{U}$ déterminées par des conjonctions finies de contraintes linéaires. Un élément A de $\mathcal{P}$ est donc un *simplexe* et il est égal à l'intersection des sous-espaces engendrés par ses contraintes linéaires. Les cas et les connaissances du domaine (`CD`) sont représentés par des simplexes.

$\mathcal{P}$ est bien stable par $\cap$, de plus pour une distance d définie à partir d'une norme sur $\mathcal{U}$ (comme c'est le cas pour les exemples 2 et 3), les éléments de $\mathcal{P}$ sont bien des fermés. On peut donc utiliser les assignation fidèles du type $\leqslant^d_\bullet$, ce que l'on fera dans la suite. Si $\mathcal{U}$ contient au moins deux éléments, une distance sur $\mathcal{U}$ ne peut pas être linéaire, mais le problème de la minimisation de certaines distances se réduit à celui de la minimisation d'une fonction linéaire.

Exemple 2
Soient $w_1, \ldots, w_n > 0$ des réels et d la distance définie sur $x, y \in \mathcal{U}$ par : $d(x, y) = \sum_i w_i|y_i - x_i|$. Pour $A, B \in \mathcal{P}$, la minimisation de d sur $A \times B$ est équivalente à celle de la forme linéaire sur $\mathcal{U}^3$: $(x, y, z) \mapsto \sum_i w_i z_i$ sous les contraintes : $x \in A$, $y \in B$ et pour tout i $z_i \geqslant y_i - x_i$ et $z_i \geqslant x_i - y_i$.

Exemple 3
La technique utilisée dans l'exemple précédant pour réduire la minimisation de la distance d à celle d'une fonction linéaire s'applique plus généralement aux fonctions convexes définies à partir d'un nombre fini de formes linéaires. C'est le cas, par exemple pour $d : (x, y) \mapsto \max_i(w_i|y_i - x_i|)$.

Un problème de programmation linéaire dont les contraintes sont satisfaisables et dont la fonction objectif est minorée sous ces contraintes admet une solution. Ainsi

avec les distances considérées plus haut, $\leqslant^d_\bullet$ satisfait $(A4)$, et d'après l'implication *1.* du théorème 2 l'opérateur $\dot{+}$ engendré satisfait les postulats $(\textit{Rév}1)$ à $(\textit{Rév}5)$.

4.3 Exemple

On reprend ici l'exemple de la section 2.1 Les requêtes sont formées de contraintes sur les quantités d'ingrédients et les solutions donnent leur quantité, un degré de liberté peut-être laissé (par exemple pour la quantité de sel). Elles sont donc aussi exprimées sous la forme de contraintes sur les quantités d'ingrédients et on ne distingue pas la partie problème de la partie solution dans les cas.

On utilise le formalisme de représentation vu dans la section précédente. $\mathcal{U} = \mathcal{U}_1 \times \ldots \times \mathcal{U}_n$ où chaque composante $i = 1, \ldots, n$ correspond à la quantité d'un ingrédient. Une variable x_i lui est associée, elle est également notée $X_{\text{[ingrédient]}}$ en remplaçant X par N, M ou V selon que l'ingrédient associé soit mesuré en unités ($\mathcal{U}_i = \mathbb{N}$), en grammes ou en millilitres ($\mathcal{U}_i = \mathbb{R}^+$). Les éléments de $\mathcal{P}$ sont des simplexes exprimés par des ensembles de contraintes linéaire sur ces variables. Un élément $A \in \mathcal{P}$ est déterminés par une conjonction de contraintes linéaires.

L'exemple de la section 2.1 devient dans $\mathcal{U} = \mathcal{U}_1 \times \ldots \times \mathcal{U}_6$:

	variable	`Srce`	`Cible`
$\mathcal{U}_1 = \mathbb{N}$	x_1/N_{tomate}	8	0
$\mathcal{U}_2 = \mathbb{N}$	$x_2/N_{\text{concombre}}$	0	·
$\mathcal{U}_3 = \mathbb{N}$	x_3/N_{poivron}	0	0
$\mathcal{U}_4 = \mathbb{R}^+$	$x_4/M_{\text{légume-fruit}}$	·	·
$\mathcal{U}_5 = \mathbb{R}^+$	x_5/M_{feta}	100	·
$\mathcal{U}_6 = \mathbb{R}^+$	$x_6/V_{\text{vinaigrette}}$	20	·

$\texttt{CD} = \texttt{CD}_{\text{légume-fruit}}$ où $\texttt{CD}_{\text{légume-fruit}}$ lie la masse de légume-fruit à la quantité des ingrédients qui en font partie :

$$\texttt{CD}_{\text{légume-fruit}} = [M_{\text{légume-fruit}} = 80 \cdot N_{\text{tomate}} + 300 \cdot N_{\text{concombre}} + 70 \cdot N_{\text{poivron}}]$$

La recette `Srce` contient donc 640 g de légumes-fruits.

La distance utilisée est de la forme $d(x, y) = \sum_i w_i |y_i - x_i|$ avec des coefficients w_i à déterminer. Selon des considérations sur les valeurs à conserver en priorité, des ordres de grandeurs peuvent être donnés pour les rapports entre les w_i. Dans l'exemple, on donne priorité à la conservation de la masse de légumes-fruits avant celle du nombre de tomates, de concombres et de poivrons. On prendra alors $w_4 \gg 80 \times w_1 + 300 \times w_2 + 100 \times w_3$ (les valeurs des variables n'étant pas normalisées, les coefficients sont nécessaires pour comparer les w_i). Ce qui donne :

$$\begin{aligned}\texttt{CibleComplété} = {} & [N_{\text{tomate}} = 0] \cap [N_{\text{concombre}} = 2] \cap [N_{\text{poivron}} = 0] \\ & \cap [M_{\text{légume-fruit}} = 600] \cap [M_{\text{feta}} = 100] \cap [V_{\text{vinaigrette}} = 20]\end{aligned}$$

4.3.1 Expression de la requête

Avec `Cible` nous avons pu exprimer la restriction à certains ingrédients sous forme d'interdiction des autres. Cependant l'expression de la demande d'ingrédient pose problème, par exemple la contrainte « avec du concombre » ne peut être exprimée sous la forme $N_{\text{concombre}} > 0$ car seules les inégalités larges sont autorisées.

L'approximation consistant à introduire une valeur arbitraire minimale n'est pas non plus satisfaisante. Par exemple avec $\texttt{Cible} = [N_{\text{concombre}} \geqslant 1]$, et un choix de w_i qui favorise la conservation de la quantité de légumes-fruits par rapport à celui de tomates, $\texttt{CibleComplété} = [N_{\text{tomate}} = 4] \cap [N_{\text{concombre}} = 1] \cap [N_{\text{poivron}} = 0] \cap [M_{\text{légume-fruit}} = 620] \cap [M_{\text{feta}} = 100] \cap [V_{\text{vinaigrette}} = 20]$. C'est-à-dire que, mis à part le concombre imposé, on conserve les tomates. On s'attendrait plutôt à ce que les tomates soient complètement remplacées par du concombre. Il y a une variété de légume-fruit dans le cas source, les tomates. Plutôt que d'en substituer quelques unes par du concombre et obtenir un mélange, on préfère conserver le fait qu'il n'y ait qu'une variété de légume-fruit, quitte à ce que ce soit du concombre et non des tomates.

Cela se traduit par l'introduction d'une valeur associée à chaque ingrédient, le nombre de variétés de cet ingrédient : $\#_{\text{[ingrédient]}}$ à valeurs dans $\mathbb{N}$. Pour un concept dans la hiérarchie des ingrédients, c'est le nombre de feuilles sous lui pour lesquelles les quantités sont non nulles. Dans `Srce`, le nombre de variétés de tomates est 1, pour les concombres et les poivrons il est de 0 et il est donc de 1 pour les légumes-fruits. Ces valeurs sont à prendre en compte lors de l'adaptation, on ajoute donc des dimensions à $\mathcal{U} : \mathcal{U} = \mathcal{U}_{\text{quantités}} \times \mathcal{U}_{\text{variétés}}$ avec $\mathcal{U}_{\text{variétés}} = \mathbb{N}^6$ et $\mathcal{U}_{\text{quantités}}$ l'espace $\mathcal{U}$ précédant. Les nouvelles variables sont reliées aux précédantes par $\texttt{CD}_{\text{variétés}}$:

$$\texttt{CD}_{\text{variétés}} = [\#_{\text{légume-fruit}} = \#_{\text{tomate}} + \#_{\text{concombre}} + \#_{\text{poivron}}] \cap \bigcap_{\text{[ingrédient]}} [\#_{\text{[ingrédient]}} = \texttt{si } (X_{\text{[ingrédient]}} > 0) \texttt{ alors } 1 \texttt{ sinon } 0 \texttt{ fsi}]$$

Avec $\texttt{CD} = \texttt{CD}_{\text{légume-fruit}} \cap \texttt{CD}_{\text{variétés}}$. Les contraintes de $\texttt{CD}_{\text{variétés}}$ ne sont pas sous forme linéaires et nous n'avons pas réussi à réduire le problème à de la programmation linéaire. Deux pistes sont envisagées, la première consiste à approcher les contraintes du type « $\#_{\text{[ingrédient]}} = \texttt{si } (X_{\text{[ingrédient]}} > 0) \texttt{ alors } 1 \texttt{ sinon } 0 \texttt{ fsi}$ » par « $\#_{\text{[ingrédient]}} \geqslant \varepsilon \cdot X_{\text{[ingrédient]}}$ » avec $\varepsilon > 0$ suffisamment petit pour que $\varepsilon \cdot X_{\text{[ingrédient]}} < 1$ pour les valeurs raisonnables de $X_{\text{[ingrédient]}}$.

La deuxième piste consiste à procéder à l'adaptation en deux temps : d'abord dans $\mathcal{U}_{\text{variétés}}$ puis dans $\mathcal{U}_{\text{quantités}}$. Remarquons déjà que les variables $\#_{\text{[ingrédient]}}$ sont majorées par le nombre de feuilles qui sont sous [ingrédient] dans la hiérarchie des ingrédients. Les projections $(\texttt{CD} \cap \texttt{Srce})_{\text{variétés}}$ et $(\texttt{CD} \cap \texttt{Cible})_{\text{variétés}}$ des contextes source et cible sur $\mathcal{U}_{\text{variétés}}$ sont donc des ensembles finis. En pratique `Srce` est souvent un singleton et `Cible` est exprimé par des contraintes sur $\mathcal{U}_{\text{variétés}}$, dans l'exemple $\texttt{Cible} = [\#_{\text{concombre}} = 1]$, $(\texttt{CD} \cap \texttt{Srce})_{\text{variétés}}$ et $(\texttt{CD} \cap \texttt{Cible})_{\text{variétés}}$ sont donc aussi des simplexes. On peut donc calculer la révision de $(\texttt{CD} \cap \texttt{Srce})_{\text{variétés}}$ par $(\texttt{CD} \cap \texttt{Cible})_{\text{variétés}}$ induite par la distance sur $\mathcal{U}_{\text{variétés}}$, ce qui donne un ensemble fini $\texttt{CibleComplété}_{\text{variétés}}$. Dans cet exemple, il s'agit d'un singleton :

variable	$\texttt{CD} \cap \texttt{Srce}_{\text{variétés}}$	$\texttt{CD} \cap \texttt{Cible}_{\text{variétés}}$	$\texttt{CibleComplété}_{\text{variétés}}$
$\#_{\text{tomate}}$	1	·	0
$\#_{\text{concombre}}$	0	1	1
$\#_{\text{poivron}}$	0	·	0
$\#_{\text{légume-fruit}}$	1	$\geqslant 1$	1
$\#_{\text{feta}}$	1	·	1
$\#_{\text{vinaigrette}}$	1	·	1

Dans le cas où $\texttt{CibleComplété}_{\text{variétés}}$ est un singleton, les contraintes qu'il entraîne sur les quantités ($\texttt{Cible}_{\text{quantités}}$) sont $X_{\text{[ingrédient]}} = 0$ lorsque $\#_{\text{[ingrédient]}} = 0$, les autres contraintes ($X_{\text{[ingrédient]}} > 0$) devant être relâchées. Dans l'exemple :

variable	$(\texttt{CD} \cap \texttt{Srce})_{\text{quantités}}$	$\texttt{CD} \cap \texttt{Cible}_{\text{quantités}}$	$\texttt{CibleComplété}_{\text{quantités}}$
N_{tomate}	8	0	0
$N_{\text{concombre}}$	0	·	2
N_{poivron}	0	0	0
$M_{\text{légume-fruit}}$	640	·	600
M_{feta}	100	·	100
$V_{\text{vinaigrette}}$	20	·	20

Si $\texttt{CibleComplété}_{\text{variétés}}$ n'est pas un singleton, les contraintes qu'il entraîne sur les quantités ne peuvent pas toujours être mises sous forme linéaire. On traite alors séparément chacun de ses éléments comme précédemment puis on agrège les résultat pour ne conserver que ceux qui minimisent la distance avec $(\texttt{CD} \cap \texttt{Srce})_{\text{quantités}}$.

Il reste à déterminer sous quelles conditions les résultats pour le problème de minimisation dans $\mathcal{U}$ (non linéaire) et ceux de la décomposition en deux problèmes de minimisation linéaire dans $\mathcal{U}_{\text{variétés}}$ puis $\mathcal{U}_{\text{quantités}}$ sont équivalents. Cette question n'a pas encore été résolue à l'heure actuelle.

4.3.2 Connaissances d'adaptation que l'on sait exprimer

Compensation. L'exemple de la salade de concombres fait intervenir de la compensation, la suppression des tomates a été compensée par l'ajout de concombres. Une compensation est entraînée par une contrainte reliant différentes variables dont une prioritaire pour la conservation. C'est le cas dans l'exemple avec N_{tomate} et $N_{\text{concombre}}$ qui sont reliés à $M_{\text{légume-fruit}}$ par $\texttt{CD}_{\text{légume-fruit}}$. Afin de conserver la valeur de la variable prioritaire, la modification de la valeur de l'une des variables moins prioritaire est compensée par une modification des autres.

Règles d'adaptation. On peut exprimer certaines règles d'adaptation sous la forme de compensation à travers l'introduction d'une variable, la variable d'état, et de règles de conservation donnant un sens à la règle d'adaptation sous la forme de la préservation de la variable d'état. Par exemple, prenons la règle qui dit que quand on remplace les tomates par du concombre dans une salade, la vinaigrette doit être remplacée par de la crème au citron. Elle peut être expliquée par le fait que les tomates libèrent du jus qui a un effet liant, ce que le concombre ne fait pas. La crème joue aussi ce rôle de liant. On introduit donc la variable V_{liant} et la connaissance associée $\texttt{CD}_{\text{liant}}$:

$$\texttt{CD}_{\text{liant}} = [V_{\text{liant}} = 10 \times N_{\text{tomate}} + V_{\text{crême citron}}]$$

Avec une distance qui donne suffisamment d'importance à la conservation de V_{liant} :

variable	$\texttt{CD} \cap \texttt{Srce}$	$\texttt{Cible}$	$\texttt{CibleComplété}$
N_{tomate}	8	0	0
$N_{\text{concombre}}$	0	·	2
N_{poivron}	0	·	0

variable	CD ∩ Srce	Cible	CibleComplété
$M_{\text{légume-fruit}}$	640	·	600
M_{feta}	100	·	100
$V_{\text{vinaigrette}}$	20	·	0
$V_{\text{crême citron}}$	0	·	80
V_{liant}	80	·	80

La vinaigrette a bien été remplacée par la crème au citron et la conservation du liant est assurée.

5 Conclusion

La formalisation de la révision proposée dans Lieber (2007) a été étendue dans ce papier à un formalisme plus général que la logique propositionnelle finie et permettant de prendre en compte des valeurs numériques. Pour cela, les postulats de Katsuno et Mendelzon pour la révision ont également été étendus à ce formalisme, ainsi que le théorème de représentation par assignement fidèle faisant le parallèle entre problème de révision et problème de minimisation. En vue d'une implantation, nous avons montré qu'avec une représentation des connaissances sous forme de simplexes et un type d'opérateur de révision défini à partir d'une distance entre interprétations, un problème de révision se réduisait à un problème de programmation linéaire. Un exemple d'adaptation de recettes de cuisine où les quantités d'ingrédients sont prises en compte a permis d'illustrer le fait que certaines connaissances d'adaptation peuvent être mises sous cette forme. En particulier, la compensation s'exprime facilement. En revanche, nous n'avons pas trouvé comment exprimer certaines requêtes linéairement. Une solution proposée consiste à décomposer l'adaptation en deux problèmes de programmation linéaire. Une implantation de cette approche est en cours. Elle s'appuie sur le solveur libre GLPK (2009). Son application au projet Taaable (Badra *et al.* (2009)) permettra d'en faire une évaluation. Cette évaluation utilisera dans un premier temps le moteur de remémoration actuellement développé. Cependant, à plus long terme, l'étude d'une remémoration plus étroitement liée à cette forme d'adaptation est envisagée.

Références

ALCHOURRÓN C. E., GÄRDENFORS P. & MAKINSON D. (1985). On the logic of theory change : partial meet functions for contraction and revision. *J. of Symbolic Logic*, **50**, 510–530.

BADRA F., COJAN J., CORDIER A., LIEBER, J. MEILENDER T., MILLE A., MOLLI P., NAUER E., NAPOLI A., SKAF-MOLLI H. & TOUSSAINT Y. (2009). Knowledge acquisition and discovery for the textual case-based cooking system WIKITAAABLE. In *Workshop Proceedings of the 8th International Conference on Case-Based Reasoning (ICCBR-09)*.

DANTZIG G. B. (1963). *Linear Programming and Extensions*. Princeton University Press.

GLPK (2009). `http://www.gnu.org/software/glpk/glpk.html/` (dernière consultation : 14 juillet 2009).

KATSUNO H. & MENDELZON A. (1991). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, **52**(3), 263–294.

LIEBER J. (2007). Application of the Revision Theory to Adaptation in Case-Based Reasoning : The Conservative Adaptation. In *Proceedings of the 7th ICCBR*.

RIESBECK C. K. & SCHANK R. C. (1989). *Inside Case-Based Reasoning*. Hillsdale, New Jersey : Lawrence Erlbaum Associates, Inc.