Apprentissage de connaissances d'adaptation à partir des feedbacks des utilisateurs

Abir-Beatrice Karami, Karim Sehaba, Benoît Encelle

Apprentissage de connaissances d'adaptation à partir des feedbacks des utilisateurs

Abir Beatrice Karami[1], Karim Sehaba[2], Benoît Encelle[1]

[1] Université de Lyon, CNRS
Université Lyon 1
LIRIS, UMR5205, F-69622, France
abir-beatrice.karami@liris.cnrs.fr
benoit.encelle@liris.cnrs.fr
[2] Université de Lyon, CNRS
Université Lyon 2
LIRIS, UMR5205, F-69676, France
karim.sehaba@liris.cnrs.fr

Résumé : Dans le cadre des systèmes adaptatifs, notre travail de recherche porte sur l'acquisition des connaissances d'adaptation à partir des traces d'interaction laissées par les utilisateurs. Les traces contiennent, entre autres, les feedbacks, positifs ou négatifs, des utilisateurs par rapport aux actions du système. Notre objectif est de définir des modèles et des outils permettant d'extraire des règles d'adaptation que le système pourra utiliser, dans son processus de décision, afin de personnaliser son comportement à l'utilisateur. Ces règles d'adaptation établissent des relations de dépendance entre certaines caractéristiques du contexte d'interaction (à savoir certains attributs de la situation d'interaction, tels que le lieu, la luminosité, etc. et/ou du profil de l'utilisateur) et le niveau de satisfaction de l'utilisateur. Pour cela, nous proposons deux algorithmes d'apprentissage. Le premier est direct est certain, dans le sens où toutes les règles générées correspondent à des contextes d'interaction déjà rencontrés par le système, mais nécessite un nombre important de traces pour converger. Le deuxième est plus rapide mais présente des risques d'erreur. En effet, cet algorithme permet de généraliser les règles d'adaptation existantes à de nouveaux contextes d'interaction (i.e. de nouvelles situations et/ou nouveaux profils d'utilisateurs). Dans cet article, nous détaillons les modèles que nous proposons pour représenter les règles d'adaptation et les traces d'interaction ainsi que les deux algorithmes d'apprentissage. Nous présentons également les évaluations que nous avons menées, par simulation et avec de vrais utilisateurs, pour valider nos contributions.

Mots-clés : Systèmes adaptatifs ; traces d'interaction ; apprentissage à partir des feedbacks utilisateurs ; extraction de connaissances d'adaptation.

1 Introduction

L'acquisition des connaissances d'adaptation dans les systèmes adaptatifs est un problème important. En effet, c'est en fonction de ces connaissances que le système s'adapte aux différentes situations auxquelles il est confronté et va faire évoluer ses connaissances. Il s'agit d'un problème complexe et les solutions proposées sont, généralement, dépendantes du domaine d'application.

La plupart des travaux sur l'acquisition des connaissances d'adaptation est basé sur le raisonnement à partir de cas (Wiratunga *et al.* (2002); d'Aquin *et al.* (2007)). Dans (Hanney & Keane (1997)), les auteurs proposent une méthode d'apprentissage des connaissances d'adaptation basée sur la comparaison. D'autres techniques, issues de la génération des recettes de cuisine, ont été développées (Ihle *et al.* (2009); Gaillard *et al.* (2012)). Dans (Gaillard *et al.* (2012)), les connaissances d'adaptation prennent la forme de *substitutions*. Ces dernières sont découvertes à travers un processus de data-mining. Un deuxième processus, piloté par un expert, est nécessaire afin de généraliser ces substitutions et les rendre utilisables dans d'autres recettes. D'autres approches sont basées sur l'analyse des traces d'interaction. Dans (Sehaba (2011)), l'auteur propose une approche permettant à différents utilisateurs de profils très différents, par rapport à

leurs compétences, capacités et/ou préférences, de partager les traces de leurs propres activités. En utilisant un processus de transformation à base de règles, les traces partagées sont adaptées à l'utilisateur cible en fonction de son profil. La génération des connaissances d'adaptation, ici, est basée sur l'application de règles prédéfinies par des experts sur les traces d'interaction.

Une autre approche basée sur les traces d'interaction est proposée (Kanda & Ishiguro (2005)). Cette approche a été appliquée sur un robot compagnon destiné aux enfants. Dans ce cadre, le système développé permet d'identifier chaque enfant et de mémoriser l'historique de ses interactions. En fonction de cet historique, le système est capable de personnaliser ses interactions à l'enfant et aussi de s'adapter à la situation. Néanmoins, les connaissances d'adaptation ici sont prédéfinies et non générées automatiquement. D'autres approches utilisant les feedbacks utilisateurs se sont développées. Dans (Knox & Stone (2009)), les auteurs proposent un framework basé sur des techniques *shaping*. Dans ces techniques, il est nécessaire que l'utilisateur observe le système afin de lui fournir un feedback sur la qualité de ses actions. Ces techniques utilisent des méthodes d'apprentissage supervisé pour générer des fonctions de renforcement. Ces fonctions sont ensuite utilisées par le système, dans son processus de décision, afin de choisir les actions les plus appropriées à l'ensemble des utilisateurs. Même si ces techniques permettent l'apprentissage de certaines tâches, elles ne sont pas adaptées à des environnements interactifs nécessitant des adaptations aux particularités et aux spécificités de chaque utilisateur.

L'objectif général de notre recherche est de développer des modèles et des outils permettant aux systèmes interactifs de s'adapter aux différentes situations et de personnaliser leurs comportements en fonction des préférences et besoins des utilisateurs. Dans ce cadre, nous nous intéressons, en particulier, à l'apprentissage des connaissances d'adaptation à partir des traces d'interaction. Ces dernières contiennent, entre autres, les feedbacks des utilisateurs. Les connaissances d'adaptation ici établissent des relations entre la satisfaction de l'utilisateur et les actions du système, le profil de l'utilisateur et/ou des informations de la situation d'interaction. Le principe de notre approche est de générer, à partir des règles d'adaptation existantes et des traces d'interaction, de nouvelles connaissances applicables à des nouveaux utilisateurs et/ou de nouvelles situations.

Notre approche a été appliquée sur un processus de décision Markovien PDM (Puterman (1994)). Les règles d'adaptation ont été intégrées dans les fonctions de récompense du PDM afin de permettre une planification adaptative. Ce travail entre dans le cadre du projet FUI Robot Populi. Ce projet, financé par le ministère de l'industrie, vise le développement d'un robot compagnon adaptatif (Karami *et al.* (2013)).

Dans la section suivante, nous présentons les concepts généraux de notre approche ainsi que la formalisation que nous proposons. Dans la section 3, nous présentons nos algorithmes d'acquisition de connaissances d'adaptation, utilisées par le système dans son processus de prise de décision. Afin de valider nos contributions, nous avons mené deux expérimentations : la première par simulation et la deuxième avec de réels utilisateurs. La section 4 décrit ces expérimentations ainsi que les résultats obtenus. La dernière section est consacrée à la conclusion et aux perspectives de ce travail.

2 Concepts et formalisations

Nous présentons dans cette section les définitions et formalisations que nous avons introduites pour représenter une situation, un profil utilisateur et les traces d'interaction.

2.1 Attributs et Contexte d'interaction

Nous définissons les *attributs* $\mathcal{AT}$ représentant un *contexte d'interaction* comme l'ensemble des informations caractérisant d'une part le profil de l'utilisateur $\mathcal{AT}^p$ (par exemple : âge, sexe, préférences, habitudes...) et, d'autre part, la situation d'interaction $\mathcal{AT}^s$ (e.g. lieu, luminosité, température...) : $\mathcal{AT} = \mathcal{AT}^p \cup \mathcal{AT}^s$. Chaque attribut $at \in \mathcal{AT}$ possède une valeur appartenant à son domaine de définition (e.g. $sexe \in \{homme, femme\}$). Une décision (qui se concrétise ensuite par une action) peut être liée à un ou plusieurs de ces attributs. Par exemple, la décision *projeter un film d'action* à un utilisateur donné va être essentiellement liée à un attribut "aime_film_d'action" du profil de l'utilisateur, s'il existe. Naturellement, d'autres attributs peuvent influer sur cette décision tels que : l'heure de la journée, le lieu, l'âge de l'utilisateur...

Dans une assistance interactive ou un système compagnon, il est très difficile, voire impossible, de définir manuellement l'ensemble des règles d'adaptation couvrant tous les contextes d'interaction (i.e. tous les profils des utilisateurs potentiels et toutes les situations possibles). C'est pourquoi, nous visons au développement de techniques d'acquisition automatique permettant de déterminer les attributs importants $\mathcal{AT}_{a_i}$ influant sur chaque décision/action $a_i \in \mathcal{A}$ (l'ensemble des décisions/actions). Ces techniques se basent sur les feedbacks (positifs ou négatifs), stockés dans des traces, des utilisateurs par rapport aux expériences passées. Nous notons $\mathcal{AT}_{a_i} \subseteq \mathcal{AT}$ les attributs importants pour a_i.

Une action adaptée doit prendre en compte *le contexte d'interaction* (i.e. la situation courante et les propriétés du profil de l'utilisateur). Les informations du contexte sont représentées par $s \in \mathcal{S}$. Ainsi,

$$\begin{aligned}\mathcal{S} &= \mathcal{AT} = \mathcal{AT}^p \cup \mathcal{AT}^s \\ &= \{(at^{p_1}, \ldots at^{p_n}, at^{s_1}, \ldots at^{s_m}) : at^{p_i} \in \mathcal{AT}^p, at^{s_i} \in \mathcal{AT}^s\}\end{aligned}$$

2.2 Traces d'interaction et règles d'adaptation

Les traces d'interaction et les règles d'adaptation ont des représentations similaires. En effet, leurs deux modèles contiennent un contexte d'interaction $s \in \mathcal{S}$, une action du système $a \in \mathcal{A}$ et une valeur $v \in [-1, +1]$. Formellement, les traces et les règles sont représentées comme suit : $\mathcal{A} \times \mathcal{S} \rightarrow v$. Elles possèdent néanmoins deux différences :

1. Dans une trace d'interaction, v est une valeur numérique représentant le feedback de l'utilisateur (feedback utilisateur suite à une action du système a). Cette valeur est fournie par une fonction prédéfinie $valeur\_feedback$, par exemple : *valeur_feedback (sourire)=1*. En revanche, la valeur de v dans une règle d'adaptation est calculée par un processus d'apprentissage (cf. section 3).
2. Alors que les attributs d'un contexte d'interaction donné s, dans une trace d'interaction, possèdent des valeurs appartenant à leurs domaines respectifs de définition, les valeurs possibles d'attributs, dans une règle d'adaptation, sont décrites à l'aide de contraintes sous forme d'expressions logiques. La figure 1 montre quelques exemples de traces d'interaction et de règles d'adaptation (le symbole * indique que n'importe quelle valeur pour l'attribut est acceptée).

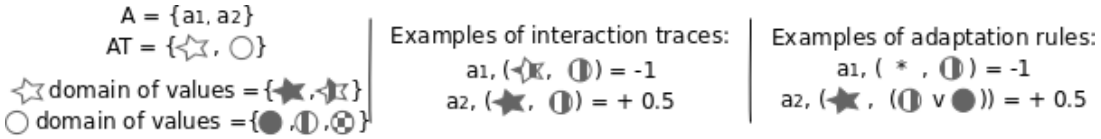

FIGURE 1 – Exemple de traces d'interaction et de règles d'adaptation

Nous précisons qu'une trace d'interaction est *incluse* dans une règle d'adaptation si les valeurs des différents attributs de la trace respectent les contraintes de la règle d'adaptation.

3 Algorithmes d'apprentissage

Dans cette section, nous présentons deux algorithmes d'acquisition de connaissances d'adaptation $\mathcal{K}$ à partir des traces d'interaction. Le premier, direct et certain, permet de générer des règles valables à tout moment. Le deuxième permet de généraliser les règles apprises afin de les appliquer à des nouveaux contextes d'interaction ; c'est-à-dire à des profils et/ou des situations inconnues. Cet algorithme, à la différence du premier, est à risque dans la mesure où il est basé sur la détection automatique des attributs importants $\mathcal{AT}_{a_i}$ pour chaque action a_i. Les deux algorithmes ont principalement comme entrée un ensemble de règles d'adaptation $\mathcal{K}$ apprises à partir des expériences passées (cet ensemble est vide initialement) et une nouvelle expérience, représentée par une nouvelle trace notée $newTrace$, que le système utilise pour mettre à jour ses connaissances d'adaptation $\mathcal{K}$. En sortie, les deux algorithmes retournent un ensemble de règles d'adaptation, éventuellement mis à jour (i.e. ajout de nouvelles règles ou modification de règles existantes dans $\mathcal{K}$).

3.1 Algorithme direct

L'idée principale de l'algorithme 1 est de mettre à jour les règles d'adaptation $\mathcal{K}$ à partir d'une nouvelle trace $newTrace$ de façon simple et directe. Comme le montre cet algorithme, plusieurs situations sont considérées. Les règles d'adaptation concernées par l'apprentissage à partir de $newTrace$ sont celles qui portent sur la même action $a \in A$. Ce qui signifie que pour une nouvelle trace $newTrace : (a_i, s) = value$ seules les règles d'adaptation $\mathcal{K}_a \subseteq \mathcal{K}$ où $a = a_i$ sont considérées. Ainsi :

1. Pour chaque $k \in \mathcal{K}_a$ où $a = a_i$: Si $newTrace$ est *incluse* dans k (Ligne 5), alors :
   (a) Si la valeur de feedback pour $newTrace$ et la valeur de k sont dans la même direction (les deux sont positives ou négatives) (Ligne 6), alors on affecte à la valeur de k la moyenne des deux valeurs (Ligne 7).
   (b) Si les deux valeurs sont de signes différents (une positive et l'autre négative), c'est-à-dire que le feedback de l'utilisateur est en *contradiction* avec une règle existante, $newTrace$ et k seront marquées (Ligne 9 :10). L'ensemble des règles marquées est alors présenté à l'expert afin de détecter les raisons de la contradiction. L'origine de cette dernière peut être le manque d'attributs dans la représentation du problème [1].

1. Par exemple un utilisateur peut changer d'avis si il pleut, d'où l'éventuel besoin d'ajouter un attribut représentant la météo.

2. Si $newTrace$ contient un nouveau contexte d'interaction, n'appartenant à aucune des règles existantes, alors $newTrace$ est ajoutée comme une nouvelle règle d'adaptation dans $\mathcal{K}$ (lignes 11 à 12).

Algorithm 1 Algorithme d'apprentissage direct

```
1: INPUT K, newTrace.
2: Output K.
3: added = false, contradiction_detected = false
4: for all k ∈ K_a do
5:     if (newTrace included in k) then
6:         if (sameDirection(newTrace.value, k.value)) then
7:             k.value = average(newTrace.value, k.value)
8:             added = true
9:         else
10:            contradiction_detected = true
11: if (¬added AND ¬contradiction_detected) then
12:     Add newTrace to K_a
```

3.2 Algorithme de généralisation

L'algorithme 2 a une approche très différente du premier. En effet, il se caractérise par sa capacité à apprendre de nouvelles règles tout en se basant sur un nombre d'expériences moins important. Pour cela, cet algorithme permet de généraliser les règles existantes afin de les appliquer à des nouveaux contextes jusqu'alors inconnus du système.

L'idée principale est de détecter, pour chaque action possible $a \in \mathcal{A}$, l'ensemble des attributs importants (relatifs à l'utilisateur et/ou à la situation) $\mathcal{AT}_a$, c'est à dire ceux dont les valeurs affectent le feedback de l'utilisateur. Les attributs non importants seront alors généralisés à "n'importe quelle valeur " et représentés par des symboles * dans les règles d'adaptation concernées (Ligne 14). Cet algorithme sauvegarde toutes les traces d'interaction. De ce fait, toutes les traces d'interaction sont utilisées, en continu, dans le processus de détection des attributs importants.

Dans les paragraphes suivants, nous expliquons à l'aide d'un exemple, (a) comment l'algorithme utilise les contradictions entre une $newTrace$ et une règle $k \in \mathcal{K}_a$ pour détecter automatiquement les attributs importants et, (b) comment détecter automatiquement les erreurs. Il s'agit des attributs détectés comme étant importants dans la première phase mais, en réalité, non importants.

Initialement, avant la réception d'une première trace $newTrace$, l'ensemble des attributs importants pour chacune des actions possibles est vide ($\mathcal{AT}_a = \emptyset, \forall a \in \mathcal{A}$), et l'algorithme généralise toutes les valeurs des attributs à * dans une nouvelle règle qui correspond à la $newTrace$ (la valeur de la règle produite est alors égale à la valeur du feedback).

Algorithm 2 Algorithme d'apprentissage par généralisation

1: **INPUT** $\mathcal{K}$, $newTrace$, $backupTraces$.
2: **Output** $\mathcal{K}, \mathcal{AT}_a$.
3: **for all** $k \in \mathcal{K}_a$ **do**
4: **if** ($newTrace$ included in k) **then**
5: **if** ($\neg sameDirection(newTrace.value, k.value)$ OR
6: $(sameDirection(newTrace.value, k.value)$ AND
7: $|newTrace.value - k.value| > \epsilon)$) **then**
8: Set $relatedBackup$ as all traces related to k and in the same direction.
9: **for all** $r \in relatedBackup$ **do**
10: **for all** $at \in \mathcal{AT}$ **do**
11: **if** ($k.at = *$ AND $r.at \neq newTrace.at$ AND $at \notin \mathcal{AT}_a$) **then**
12: $\mathcal{AT}_a = \mathcal{AT}_a \cup at$
13: **for all** $k \in \mathcal{K}_a$ **do**
14: Generalize unimportant attributes in k.
15: Specialize important attributes in k from $backupTraces$
16: $k.value = fct(newTrace.value, k.value)$
17: **for all** $at \in \mathcal{AT}_a$ **do**
18: **if** ($\neg$ ConfirmedImportant(at)) **then**
19: $\mathcal{AT}_a = \mathcal{AT}_a - at$.
20: **if** ChangeInImportantAttributes **then**
21: repeat lines

3.2.1 Extraction des attributs importants par traitement des contradictions

Dans les lignes 4 à 7, l 'algorithme 2 examine s'il existe une contradiction entre $newTrace$ et chaque règle $k \in \mathcal{K}_a$. Une contradiction est détectée si $newTrace$ est *incluse* dans k et si leurs valeurs sont opposées (i.e. l'une est positive et l'autre négative) ou sont trop éloignées (la valeur absolue de leur différence est supérieure au seuil ϵ). L'algorithme traite chaque contradiction en vue d'extraire les attributs importants liés à une action a (lignes 8 à 12) comme suit : premièrement, un ensemble non-vide $relatedBackup$ est constitué des traces d'interaction pré-existantes reliées à k et de même direction (i.e. valeurs de même signe). Une trace d'interaction est reliée à k si elle concerne la même action et qu'elle est *incluse* dans k. Ensuite, l'algorithme extrait pour chaque trace de $relatedBackup$ les attributs qui peuvent être source de contradiction avec k (valeurs d'attribut différentes) et les marque comme étant importants. Dans l'exemple donné en Figure 2, après réception d'une deuxième trace, une contradiction est détectée entre cette trace et la règle existante dans $\mathcal{K}_{a_1}$. Cette contradiction est détectée parce que la trace est incluse dans la règle mais a une valeur opposée. Comme indiqué en rouge sur la figure, deux attributs importants sont alors détectés (le cercle et le carré) suite au test défini en ligne 11.

Dans les lignes 13 à 16, l'algorithme commence par généraliser tous les attributs non importants à (*) pour toutes les règles $k \in \mathcal{K}_a$. Ensuite, il spécialise, au regard des attributs nouvellement détectés comme importants, les valeurs des attributs de k en ré-examinant l'ensemble des $backupTraces$. Une règle est alors ajoutée pour chaque trace d'interaction incluse dans k. Au final, la valeur de chaque règle k correspond à la moyenne des valeurs des traces d'interaction

de $backupTraces$ qui sont *incluses* dans k.

3.2.2 Identification des attributs faussement détectés comme importants

L'algorithme peut détecter des attributs comme étant importants alors qu'ils ne le sont pas en réalité. De ce fait, la ligne 18 confirme l'importance réelle de l'attribut.

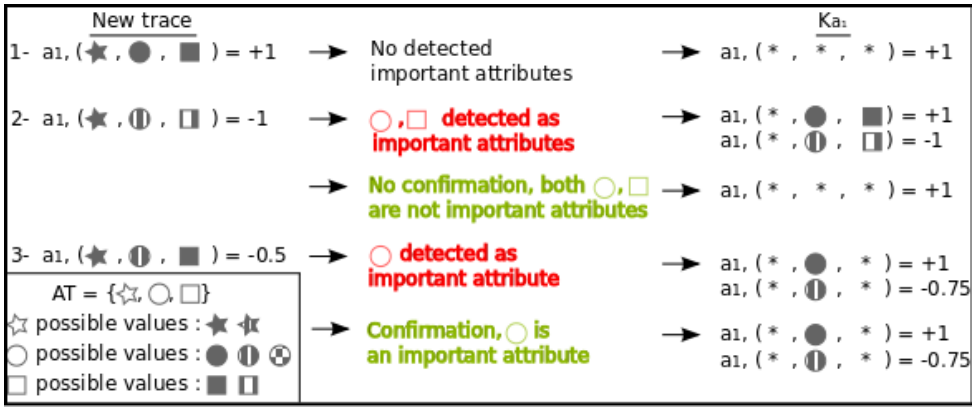

FIGURE 2 – Exemple de détection d'attributs importants : Algorithm 2.

Dans la négative, l'attribut est enlevé de l'ensemble des attributs importants $\mathcal{AT}_a$.

Plus précisément, un attribut important at est confirmé si la condition suivante est vraie pour chaque $k \in \mathcal{K}_a$: il existe au moins une autre règle $k_i \neq k$ où pour tous les autres attributs importants $at_i \neq at$ la valeur de l'attribut dans k_i est égale à sa valeur dans k et les valeurs de ces deux règles sont opposées ou trop éloignées.

Dans l'exemple donné en Figure 2, après réception de la deuxième trace, l'attribut cercle n'est pas confirmé comme étant important (en vert) parce qu'il y a un autre attribut important (le carré) pour lequel il n'y a aucune autre règle ayant la même valeur d'attribut (i.e. le même carré) avec une valeur opposée ou éloignée. De la même manière, l'attribut carré n'est pas confirmé comme étant important. Après réception d'une troisième trace, le cercle est confirmé comme important car la condition est satisfaite (sachant qu'il n'y a pas d'autres attributs importants).

4 Résultats expérimentaux

Dans cette section nous présentons les résultats obtenus après application des deux algorithmes et montrons la capacité de l'algorithme par généralisation à détecter les attributs importants pour chaque action $a \in \mathcal{A}$. Nous avons conduit deux expérimentations. La première a été effectuée par simulation et la deuxième avec des utilisateurs réels.

Ces expérimentations ont été menées dans le cadre du projet FUI Robot Populi. Ce projet vise le développement d'un robot compagnon adaptatif (Karami *et al.* (2013)).

4.1 Expérience en environnement simulé

4.1.1 Scénario

Pour cette expérience, nous considérons une activité de projection de vidéos à l'utilisateur. Cette activité peut être réalisée en suivant les 6 étapes suivantes : sélection 1/d'un pièce pour la projection (chambre principale, chambre secondaire, salon, cuisine), 2/d'une durée (épisode, film), 3/d'un type (dessin animé, manga, science fiction, drame, sport, spectacle, histoire, comédie), 4/d'un volume sonore (faible, moyen, fort), 5/d'une luminosité (faible, moyenne, forte) pour finalement 6/projeter le vidéo. Pour chaque étape, le robot doit décider des actions à réaliser pour que l'utilisateur soit au final satisfait. Dans ce scénario, les attributs représentant l'utilisateur $\mathcal{AT}^p$ sont sa tranche d'âge (enfant, adolescent, adulte) et son sexe (homme, femme). Les attributs modélisant la situation $\mathcal{AT}^s$ sont le niveau de bruit (faible, moyen, élevé), la période de la journée (matin, midi, après-midi, soirée, nuit), la luminosité (faible, moyenne, élevée) et l'étape actuelle dans l'activité.

4.1.2 Procédure

La procédure suivie pour l'évaluation des algorithmes est une boucle, constituée des éléments suivants : (1) Re/calculer la politique de décision du système (politique du MDP). (2) Générer n traces. (3) Evaluer les actions décidées pour chacun des n traces. (4) Extraire et mettre à jour les règles d'adaptation au regard des n traces (mise à jour de la fonction de récompense du MDP). (5) Réitérer ce processus jusqu'à atteindre 2000 traces.

La génération des traces (*i.e.* étape 2) est effectuée par simulation en employant d'une part la politique du MDP pour générer l'action du système et, d'autre part, quelques règles de préférence prédéfinies générant des feedbacks utilisateur. Ces dernières sont prédéfinies par rapport à une action donnée et reliées à certaines valeurs d'attribut du contexte d'interaction (*e.g.* l'action de sélection du type de vidéo dépend de la tranche d'âge et du sexe).

Le comportement du système est généré en utilisant une politique de processus de décision markovien (MDP) calculée en employant un algorithme "value iteration" sur le modèle du MDP (Puterman (1994)). Les états du MDP représentent les contextes d'interaction possibles (i.e. les attributs du profil utilisateur $\mathcal{AT}^p$ et de la situation $\mathcal{AT}^s$). L'ensemble des actions du MDP représente les actions qu'il est possible d'effectuer dans les différentes étapes de l'activité. La fonction de transition change de manière déterministe l'état en fonction de l'action. Une fonction de récompense par défaut est donnée pour respecter l'ordre des étapes permettant de réaliser l'activité (se déplacer dans une pièce, choisir la durée de la vidéo, etc.). La première politique du MDP permet de respecter l'ordre des étapes de l'activité sans inclure des règles d'adaptation.

Dans un premier temps, les traces générées ont à chaque fois respecté les règles de préférence prédéfinies. Il n'y avait pas par conséquent, pour un même contexte d'interaction, de feedbacks utilisateur opposés. Cependant, de telles ambiguïtés (des réactions utilisateur différentes dans un même contexte d'interaction) peuvent exister dans le réel. Par exemple, la plupart mais pas la totalité des adultes hommes aime regarder les émissions sportives. Pour cette raison, dans un second temps, nous avons généré des traces avec une certaine probabilité d'ambiguité (*e.g.* 2% d'ambiguité : une trace est générée avec 2% de chance d'avoir un valeur de feedback utilisateur opposée à celle donnée dans les règles de préférence prédéfinies).

4.1.3 Résultats

Nous avons évalué les deux algorithmes à l'aide de la même base de contextes d'interactions générés (situations et profiles utilisateur générés au hasard). Nous avons suivi la procédure décrite auparavant avec $n = 100$ traces. Après chaque itération (chaque $n = 100$ traces), nous avons calculé le nombre d'actions suivies d'un feedback utilisateur négatif.

Les résultats de la Figure 3 montrent les courbes de convergence des deux algorithmes d'apprentissage vers un comportement adaptatif et personnalisé optimal, i.e. sans actions négatives. Les deux algorithmes ont été capables d'apprendre complètement (à 100%) les règles de préférence prédéfinies. Nous avons mené une expérimentation avec des règles de préférence dépendantes au maximum de deux attributs importants (Figure 3, à gauche), et une autre avec un maximum de trois attributs importants (Figure 3, à droite). Dans les deux cas, les attributs importants prédéfinis ont été appris au cours des expérimentations.

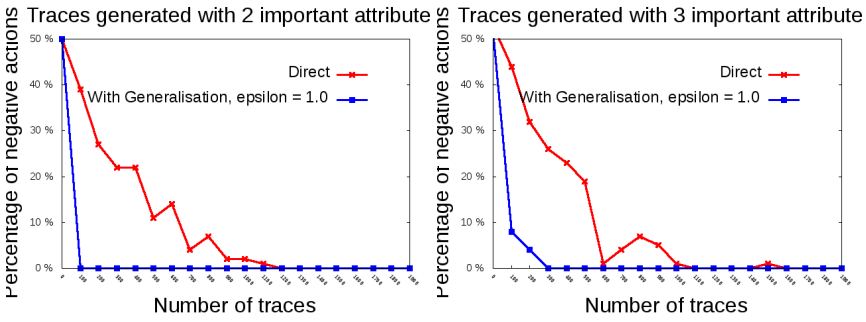

FIGURE 3 – Résultats : traces générées avec deux attributes importants (à gauche) et trois attributs importants (à droite).

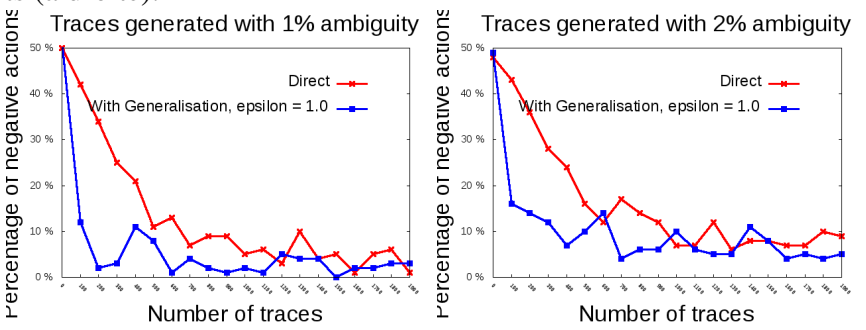

FIGURE 4 – Résultats : traces générées avec 1% d'ambiguité (à gauche) et 2% d'ambiguité (à droite).

Dans la Figure 4, nous exposons les courbes de convergence des deux algorithmes d'apprentissage pour des traces générées à 1% d'ambiguité (à gauche) et à 2% d'ambiguité (à droite). Les résultats montrent que les deux algorithmes n'arrivent pas à converger (avec moins de 2000 traces). Cependant, l'algorithme avec généralisation tend à converger plus rapidement (suite aux 500 premières traces).

4.2 Expérience avec de réels utilisateurs

Dans cette seconde expérience, des utilisateurs réels ont fourni leurs feedbacks suite aux actions du robot. Les traces ont été collectées et employées pour évaluer les algorithmes. Une activité de sélection de menu dans un restaurant à l'aide d'un Robot a été choisie pour cette expérience. Un profil utilisateur est représenté à l'aide de trois attributs : sexe (homme, femme), végétarien (oui/non) et diabétique (oui/non). La situation est représentée à l'aide des attributs suivants : moment du repas (midi, soir) et la saison (hiver ou été).

Nous avons développé une interface permettant à l'utilisateur d'introduire son profil et de préciser la situation. Après avoir donné quelques consignes à l'utilisateur et lorsqu'il est prêt à effectuer sa commande de menu, des questions sont posées de manière séquentielle pour déterminer les choix de l'utilisateur. Ces choix sont ensuite enregistrés dans sa commande afin d'être vérifiés et confirmés.

L'expérience a été réalisée avec 25 sujets adultes (14 hommes et 11 femmes). Face à la difficulté d'avoir des sujets diabétiques et/ou végétariens, chacun des sujets a effectué 4 fois l'expérimentation (i.e. 4 commandes passées au robot). Pour la première commande, le sujet a indiqué son profil réel et pour les trois autres, le système a pour chaque commande indiqué un "faux" profil devant guider ses choix, ceci dans le but de couvrir, pour chaque sujet, l'ensemble des profils possibles (4 cas : végétarien/diabétique, non végétarien/non diabétique, végétarien/non diabétique et non-végétarien/diabétique. Les valeurs des attributs représentant la situation ont été tirées au hasard.

Ces différentes situations ont chacune été distribuées de manière égale en fonction du sexe de l'utilisateur (i.e. autant d'hommes que de femmes ont passé des commandes dans une même situation). Un sujet donné a eu la même situation pour ses quatre commandes.

		M/F	Veg./Non-Veg.	Diab./Non-Diab.	Lunch/Dinner	Summer/Winter
Appetizer + Main Dish	Yes %	52/43	46/50	82/14	52/44	52/44
Appetizer + Dessert	Yes %	0/9	6/2	2/6	0/8	6/2
Main Dish + Dessert	Yes %	11/20	16/14	4/26	15/15	19/12
Appetizer + Main Dish + Dessert	Yes %	38/27	32/34	12/54	33/33	23/42
Green Salad	Yes %	60/63	98/26	60/62	59/64	62/61
Salmon Salad	Yes %	26/23	2/47	27/22	20/30	23/26
Chicken Salad	Yes %	14/14	0/28	12/16	22/7	15/13
Meat Dish	Yes %	30/32	0/61	27/36	33/29	31/31
Vegetables Dish	Yes %	70/68	100/39	73/64	67/71	69/69
Fruits	Yes %	59/68	63/64	100/56	78/52	65/62
Cake	Yes %	41/32	37/36	0/44	22/48	35/38
Red Wine	Yes %	18/9	6/22	4/24	2/25	17/12
White Wine	Yes %	9/5	12/2	2/12	4/10	8/6
Beer	Yes %	7/5	6/6	0/12	8/4	4/8
Only Water	Yes %	66/82	76/70	94/52	85/62	71/75

TABLE 1 – Résultats du premier algorithme (direct) : les réponses des utilisateurs sont catégorisées par valeur d'attribut et exprimées sous forme d'un pourcentage de réponses positives.

Nous présentons dans la Table 1 les pourcentages de réponses positives des utilisateurs (Yes) pour chaque question posée (questions en colonne 1). La table présente ces pourcentages pour les deux valeurs possibles d'un attribut (en colonne, de la colonne 3 à 5). Par exemple, si nous considérons toutes les commandes passées par des hommes pour lesquelles un plat avec viande

Appetizer + Main Dish	diabetes, daytime
Appetizer + Dessert	vegetarianism
Main Dish + Dessert	daytime, diabetes
Appetizer + Main Dish + Dessert	sex, season
Green Salad	vegetarianism, diabetes
Salmon Salad	-
Chicken Salad	diabetes
Meat Dish	vegetarianism, season
Vegetables Dish	-
Fruits	vegetarianism, daytime
Cake	-
Red Wine	sex
White Wine	vegetarianism, diabetes, daytime, season
Beer	diabetes, season
Only Water	-

TABLE 2 – Attributs détectés importants par l'algorithme avec généralisation.

a été proposé (i.e. Meat dish), dans 30% de ces commandes la réponse a été affirmative (le sujet a répondu oui à la proposition d'un plat avec viande).

La Table 2 présente les attributs qui ont été détectés comme importants par l'algorithme avec généralisation pour chaque question posée (i.e. action du robot). Notons ici l'importance des attributs indiquant si l'utilisateur est végétarien ou diabétique, ce qui semble plutôt logique. Par exemple, l'attribut indiquant si l'utilisateur est végétarien ou non a été détecté important pour l'action " Proposer une salade verte" (qui a été choisie par 98% des sujets végétariens, cf. Table 1).

5 Conclusion

Cet article présente des méthodes d'apprentissage qu'un système adaptatif peut employer pour apprendre les préférences de ses utilisateurs à l'aide de leurs feedbacks. Les expériences conduites (par simulation et avec des utilisateurs réels) montrent les capacités de ces méthodes en terme d'extraction de connaissances d'adaptation, afin de personnaliser le comportement d'un système à ses utilisateurs. L'algorithme par généralisation est capable non seulement de traiter des ambiguïtés (feedbacks contradictoires pour un même contexte d'interaction, donnés à des moments différents), mais aussi de déterminer les caractéristiques d'un contexte d'interaction influant sur les feedbacks des utilisateurs.

Comme perspectives, nous aimerions travailler sur la mise à jour des profils utilisateurs en analysant les traces d'interaction (*e.g.* en détectant les préférences personnelles). Nous aimerions également étudier la convergence de l'apprentissage par rapport à la dimension d'un problème (*i.e.* être en capacité de quantifier le nombre de traces nécessaires à un apprentissage de qualité par rapport à la complexité d'un contexte d'interaction donné).

Références

D'AQUIN M., BADRA F., LAFROGNE S., LIEBER J., NAPOLI A. & SZATHMARY L. (2007). Case base mining for adaptation knowledge acquisition. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, p. 750–755, San Francisco, CA, USA.

GAILLARD E., NAUER E., LEFEVRE M. & CORDIER A. (2012). Interactive Cooking Adaptation Knowledge Discovery for the TAAABLE Case-Based Reasoning System. In *Workshop Cooking with Computer - Conférence ECAI 2012*.

HANNEY K. & KEANE M. (1997). The Adaptation Knowledge Bottleneck : How to Ease it by Learning from Cases. In *Proceedings of the International Conference on Case-Based Reasoning.*

IHLE N., NEWO R., HANFT A., BACH K. & REICHLE M. (2009). Cookiis - A Case-Based Recipe Advisor. In S. J. DELANY, Ed., *Workshop Proceedings of the 8th International Conference on Case-Based Reasoning*, p. 269–278, Seattle, WA, USA.

KANDA T. & ISHIGURO H. (2005). Communication robots for elementary schools. *Proceedings of AISB'05 Symposium Robot Companions : Hard Problems and Open Challenges in Robot-Human Interaction (Hatfield Hertfordshire)*, p. 54–63.

KARAMI A. B., SEHABA K. & ENCELLE B. (2013). Towards Adaptive Robots based on Interaction Traces : A User Study . In *The 16th International Conference on Advanced Robotics, ICAR 2013.*, p. 1–6.

KNOX W. B. & STONE P. (2009). Interactively shaping agents via human reinforcement : the tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, K-CAP '09, p. 9–16, New York, NY, USA : ACM.

PUTERMAN M. L. (1994). *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. New York, NY, USA : John Wiley & Sons, Inc., 1st edition.

SEHABA K. (2011). Partage d'expériences entre utilisateurs différents : adaptation des modalités d'interaction. In *IC 2011 - 22èmes Journées Francophones d'Ingénierie des Connaissances*, p. 639–655.

WIRATUNGA N., CRAW S. & ROWE R. (2002). Learning to adapt for case-based design. In S. CRAW & A. D. PREECE, Eds., *ECCBR*, volume 2416 of *Lecture Notes in Computer Science*, p. 421–435 : Springer.