2013 IEEE 25th International Conference on Tools with Artificial Intelligence

Adaptive and Personalised Robots - Learning from Users' Feedback

Abir B. Karami
Universit de Lyon, CNRS
Universit Lyon 1
LIRIS, UMR5205, F-69622, France
Email: abir-beatrice.karami@liris.cnrs.fr

Karim Sehaba
Universit de Lyon, CNRS
Universit Lyon 2
LIRIS, UMR5205, F-69676, France
Email: karim.sehaba@liris.cnrs.fr

Benoît Encelle
Universit de Lyon, CNRS
Universit Lyon 1
LIRIS, UMR5205, F-69622, France
Email: benoit.encelle@liris.cnrs.fr

***Abstract*—Service robots have become increasingly important subjects in our lives. However, they are still facing problems like adaptability to their users. While major work has focused on intelligent service robots, the proposed approaches were mostly user independent. Our work is part of the FUI-RoboPopuli project, which concentrates on endowing entertainment companion robots with adaptive and social behaviour. In particular, we are interested in robots that are able to learn and plan so that they adapt and personalize their behaviour according to their users. Markov Decision Processes (MDPs) are largely used for adaptive robots applications. However, one challenging point is reducing the sample complexity required to learn an MDP model, including the reward function. In this article, we present our contribution regarding the representation and the learning of the reward function through analysing interaction traces (*i.e.* the interaction history between the robot and their users, including users' feedback). Our approach permits to generalise the learned rewards so that when new users are introduced, the robot may quickly adapt using what it learned from previous experiences with other users. We propose, in this article, two algorithms to learn the reward function. The first is direct and certain; the robot applies with a user what it learned during interaction with same kind of users (*i.e.* users with similar profiles). The second algorithm generalises what it learns to be applied to all kinds of users. Through simulation, we show that the generalised algorithm converges to an optimal reward function with less than half the samples needed by the direct algorithm.**

***Keywords*-Adaptive and personalised robots; Learning from users feedback; Markov Decision Processes MDPs.**

I. Introduction

Service companion robots are becoming more than experimental and increasingly practical inside our houses and assisted living facilities. Studied applications whether in laboratories or factories are numerous and various, like entertainment and educational robots [1], or social and nursing companion robots to elderly and disabled people [2], [3]. Few of these applications can afford to be user independent, *i.e.* the robot considers all users as similar entities and do not distinguish each user as an individual person. Therefore, when developing such robots we cannot deny social implications and users' expectations, *i.e.* we must take into account the fact that the robot's environment includes several kinds of users with different age, preferences, needs, *etc.* and the robot's behaviour should take into account these particularities.

Recently, more interest has been oriented towards user dependent approaches. One approach concentrates on adapting by analysing the interaction history with each user and adjusting the robot behaviour in consequence by proposing new and appropriate interactions [4]. Another approach concerns a rehabilitation robot that learn how to maximise the performance of users by adapting the personality of the robot according to the personality of the user [5]. In these approaches, the robot does not learn the preferences of the user, however, it matches certain behaviour parameters (speed, voice, song to sing, ...) with the user's personality (extroversion, introversion) or with the history of interaction (sing a song after being introduced). However, it is important for personalised service robots that the robot adapts itself by connecting the most suitable behaviour to some user profile particularities which is what we will use in our approach to generalise the learned behaviour to other users. In [6], the authors propose a framework where a user can shape the robot directly with good and bad signals in order to learn how to adapt to his preferences (as for a domestic dog). However, the approach is not proposed to learn personal preferences nor to handle environments including several users with different preferences.

In this article, we focus on designing an adaptive companion robot that is able to adapt and personalise its behaviour according to the situation and its user. We focus in particular on two questions:

1) How the robot can adapt its behaviour according to the situation (day time, noise, ...) and the user, taking into account his profile (age, preferences, habits, ...)? For this problem, our approach is based on MDP models where all needed information about the user profile and the activity are integrated in the MDP state. A correct MDP reward function generates an adapted and personalised robot behaviour which leads us to the second question;
2) How the robot can learn and evolve its knowledge about its users and their preferences? For this problem, our approach consists in analysing the traces of interaction between the robot and the users in order to determine the reward function. The interaction traces include the robot's actions, some ambient parameters (*i.e.* light level, noise level, *etc.*) and the users' feedback. This feedback

1082-3409/13 $31.00 © 2013 IEEE
DOI 10.1109/ICTAI.2013.98

is analysed by the robot to learn more about the users' preferences and update the reward function. Our approach also generalises the rewards by learning the dependence between the robot behaviour and certain information about the user and/or the activity, this helps the robot to adapt faster with new users.

In the following: we give more details about the context of our research project in Section II. We then present the state of the art and the proposed approaches in the related domains, their limitations and our position regarding these limitations in Section III. We contribute with our proposition of a general architecture and knowledge representation for an adaptive and personalised robot system in Section IV. In Section V we present the robot planning system based on an MDP model and in Section VI we contribute with two algorithms to learn the reward function that will help the robot planner in its adaptation process. The first algorithm learns the reward function in a direct and certain way. Using the first algorithm, the robot uses what it learned with a user to adapt to same kind of users (*i.e.* users with same profile). The second algorithm learns the connection between the most suitable behaviour and certain user profile particularities or environmental information. This connection permits to generalise what the robot learns and applies it to all kinds of users. In Section VII we present our experimental method and show the results by comparing the quality of generated policies during convergence. We show that the generalised algorithm converges to an optimal reward function with less than half the samples needed for the direct algorithm. Finally, we conclude and present our future work.

II. Robot Proposes Adaptive and Personalised Interactions

Our work is part of a research project called RobotPopuli, financed by the French ministry of industry. The project is interested in endowing the platform EMOX (EMOtion eXchange) of Awabot Corporation[1] with adaptive and social behaviour. The robot shown in Figure 1 is a turtle bot composed of a camera, laser range-finder, pico projector, micro and speaker. The current working-on version of the robot works under Linux system and uses ROS as software framework.

Fig. 1: EMOX, the augmented robot (left). EMOX, projecting a cartoon movie on the wall (right).

The idea behind the RoboPopuli project is to have a domestic entertainment companion robot that proposes activities to house occupants. The proposed activities and the way of doing them should respect the user preferences. We concentrate on activities that concern one user at a time or at most one principle user. We refer by *user*, the principle person who is sharing the activity with the robot and *potential users*, users that are not sharing an activity with the robot at the moment but have-been/might-be in the past/future. Activities can be launched manually by the user (through tablet or smart-phone interfaces for example) or by proposition from the robot.

As an illustrative example, let us consider the *activity* of projecting a video for a user (Figure 1). There are many details to take into account for such an activity. For example, what video to project: a movie, an episode, a comedy or a cartoon? what level of brightness and volume to set? should the robot propose to project the video in the living room or in the bedroom? All these details that we will call *parameters* of the activity should be set by an adaptive and personalised robot in a way that respect the activity and environment situation, the user preferences, in addition to some general rules (*e.g.* rules set by the parents for their children). Users' preferences can be general *e.g.* male adults like to watch sport or personal *e.g.* user A wants its waking-up alarm at 7 a.m.

The problem is that users preferences regarding the activity are not initially known by the robot. We want to propose an approach to help learning these preferences by analysing users' profiles and the interaction traces including users' feedback. The source of feedback can be different *e.g.* face expressions, gestures, vocal expressions, *etc.* The robot gathers/builds its knowledge through the interaction traces analysis and uses it to generate a reward function that will lead through an MDP to decisions that takes into account the learned users preferences.

III. Related Work

Most approaches for adaptive robots are based on classic formal models, like Markov models. For example, the approach proposed for Pearl-Nursebot is based on Partially Observable Markov Decision Process (POMDP) [7], [3]. A POMDP is a model to calculate optimal measures of control in uncertain environments. In order to decrease the complexity of solving the POMDP, they proposed a hierarchical POMDP based on partitioning the action space. Pearl shares the daily lives of elderly people and can help by doing actions of several categories: communication actions (*e.g.* recall to take medication or give forecast information), movement actions (*e.g.* guide an elderly somewhere), other diverse actions (*e.g.* charge battery or wait). The POMDP policy selects the best action corresponding to the current situation. Another approach based on POMDPs is proposed for adaptive decision model that adapts the actions of the robot to correspond to the estimated need of the human [8], [9]. The robot observes human actions and calculates a probability of his possible intentions and decides the appropriate interaction type (assistance, cooperation, collaboration). To overcome the complexity of solving the POMDP model, they proposed a solution based on dividing the intention estimation problem which uses a Hidden Markov Model (HMM) from the decision making which uses an MDP model. In these approaches, the robot adapts its actions to the

[1] www.awabot.com

situation and to the inferred need of a user, however, it does not propose personalised behaviour according to user profile.

Other approaches aim to learn the best actions in order to maximise certain interaction results using reinforcement learning methods. A system is proposed for an assistant robot to help through rehabilitation exercises [5]. The system learns how to adapt its behaviour and personality through three principle parameters: distance of interaction, movement speed and verbal communication (volume and speed of speech). The authors use a Policy Gradient Reinforcement Learning (PGRL) algorithm and the principal objective of the adaptive behaviour is to optimise these three parameters and adapt them to the personality of the user in order to maximise his performance (number of exercises in certain period of time). In such approaches, users feedback (performance) are used in the process of learning adaptive behaviour. However, the exact information about the users that are related to the robot decision (*e.g.* personality) are specified in advance of the learning process.

Some approaches are based on rules. In this category, a system is proposed for "Robovie" that is used at schools to help as companion to children [4]. The robot can identify the students and memorise the history of interaction with each of them. Therefore, it is able to personalise its interactions with each student and adapt depending on the history of interactions. The control model executes sequentially some activities depending on the actual situation. The order of execution is predefined as rules. The architecture of the robot permits to memorise all previous interactions with all students and follow their level of advancement. The proposed model allows a robot to propose personalised activities. However, it is based on predefined knowledge about adaptation and does not learn them by learning the preference of each user.

Several approaches uses a human feedback to learn a certain behaviour. [6] proposes a framework called Training an Agent Manually Via Evaluative Reinforcement (TAMER) based on shaping techniques. Shaping requires that a person observes the agent's actions and send a feedback signal as judgement on the action quality. Using supervised learning techniques to model a person's reinforcement function, the learned model is used to choose actions that should maximise the reinforcement. This approach can be used to teach how to adapt to user's preferences. However, it is not proposed for environments including several users with different preferences.

For personalised robot behaviour in multi users applications, it is difficult to have predefined knowledge for all possible situations and all possible users. We are interested in proposing a solution that helps the robot to learn users preferences and adapt to them by analysing, online, their feedback. In addition, how to learn the relation between these preferences and particular information in users' profiles in order to be able to generalise the adaptation capability to new users with no history of interaction with the robot.

IV. General Architecture and Knowledge Representation

In this section, we detail the general architecture of the robotic system represented in Figure 2. The system input contains observations concerning users including their **Feedback** and the environment including potential users. The system output is the robot's actions. The system includes 4 knowledge bases: **Interaction Traces**, **Activities**, **Users Profiles** and **Learned Rewards** (detailed in this section) and two main processes: **Decision Planning** and **Learning and Knowledge Extraction** (detailed in Section V and VI respectively).

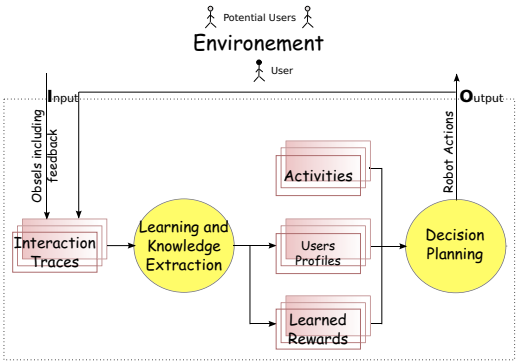

Fig. 2: General Architecture

We based our model of interaction traces on the model defined by [11]. Generally speaking, a trace is a set of temporally situated **obsels** (OBServed ELementS). Obsels are generated during interactions between the system (*e.g.* the robot) and the environment (including the users). Each obsel has a type, defined by the trace model, and might be connected (in relation) with other obsels from the same trace. Formally, the obsel has a subject (*e.g.* the user, the robot, *etc.*), and a set of attributes/values that characterises the observed event. The trace model defines the obsel types and the relations between them in the trace.

We denote input obsels related to the user $\mathcal{O}_u$, the environment and other potential users $\mathcal{O}_e$. The robot actions are integrated in the trace as output obsels $\mathcal{O}_r$. We note the set of obsels: $\mathcal{O} = \mathcal{O}_u \cup \mathcal{O}_e \cup \mathcal{O}_r$. The basic part of user related obsels $\mathcal{O}_u$ is his feedback which will be analysed to learn users preferences and how to adapt to them.

Each user is represented by a **User Profile** $\mathcal{P}$. This profile contains all information concerning the user that might be useful in the adaptive decision process. All information are represented as a set of attributes $\mathcal{B}_p$ and their values [12], [13]. Each attribute x and its domain of values $\mathcal{D}_x$ are predefined by an expert according to the application. The attributes' values of a user profile can filled by the user himself, a developer or extracted automatically by analysing the interaction traces with the user (Section VIII) [14].

Definition 1: Each profile $p \in \mathcal{P}$ can be represented as a function that associates each attribute $x \in \mathcal{B}_p$ to its value: $v^p : x \rightarrow \mathcal{D}_x \cup null$.

A profile can eventually represent the day schedule of the user. Attributes values can be static (*e.g.* name) or dynamic (*e.g.* day schedules).

Example 1: In addition to the profile id, a profile can include attributes like age with values like: child, teenage or adult; and gender with values: female or male.

There are many work in the literature on the representation of **Activities**, in particular, in the domain of human activity recognition. Classical models are mostly used like Hidden Markov Models (HMMs) [15], Petri networks [16], and Dynamic Bayesian Networks (DBNs) [17].

Our robot shares its activity with a user. The set of possible activities is predefined by $\mathcal{AC} = \{ac_1, \ldots, ac_i, \ldots, ac_m\}$. We chose to represent the activities with an HMM since that we use MDP models for the robot action planning during activities (Section VI).

Definition 2: An activity ac_i is represented by a tuple: $\langle id_{ac}, HMM_{ac}\rangle$:

- id_{ac} : the id of the activity.
- $HMM_{ac} = \langle \mathcal{Z}, \mathcal{B}_{ac}, \mathcal{A}\rangle$:
  - $\mathcal{Z} = \mathcal{O}_{ac}$, where $z \in \mathcal{O}_{ac} \subseteq \mathcal{O}$ is an obsel related to the activity ac concerning the robot, the user and the environment.
  - The set of attributes related to the activity $\mathcal{B}_{ac} = \{b_1, \ldots, b_i, \ldots, b_k\}$: an attribute $b_i \in \mathcal{B}_{ac}$ has a value $v^{ac} : b \rightarrow \mathcal{D}_b \cup null$.
  - $\mathcal{A}$ is the matrix of transitions where $a_{ijz} = P(b_t = j | b_{t-1} = i, z)$ is a probability of transitioning to a new attribute value knowing the previous attribute value and the obsel z.

The activity progress flow is represented in the transition function and triggered by the received obsels. If the activity is achieved in phases, the transition function can represent the change in attributes values in a way that respects the phase sequence to achieve the activity.

Example 2: For a robot activity of projecting a video to the user, only the robot is active to achieve the task. The task can be achieved in phases. $\mathcal{Z}$ will gather all robot's actions for all the phases. For example, starting by moving phase, the robot will move to the right room of the house: living room, kitchen, master bedroom or children bedroom. Then, choosing the length of the video phase: film or episode. Then choosing the type of the video phase: ferry cartoon, fantastic cartoon, science fiction, drama, sports, show, historic or comedy. Then setting the sound level phase: low, medium or high. Then setting the brightness level phase: low, medium or high. And finally, start projecting the video. The attributes of the activity $\mathcal{B}_{ac}$ might include for example the level of noise in the environment: low, medium or high; the day time: morning, noon, afternoon, evening or night; and the brightness in the room: low, medium or high; and finally the actual phase in the activity: the selection of room, video length, video type, volume, and brightness. Therefore $\mathcal{B}_{ac} = \{noise, daytime, brightness, phase\}$. The matrix of transitions represents the probability of changing the values of any attribute knowing that at any time step the robot can do one action from $\mathcal{Z}$.

In our model, an **Interaction Trace** includes the sequence of input/output during *one activity* between the robot and a user.

Definition 3: A trace $\mathcal{T}$ is represented by a tuple : $\langle id_t, id_p, id_{ac}, o_1, \ldots o_i, \ldots, o_n\rangle$ where,

- id_t, id_p, id_{ac}: the trace id, the user id and the id of the activity represented in this trace, respectively.
- $o_1, \ldots o_i, \ldots, o_n$: the sequence of obsels related to the activity represented in this trace, where o_i: $\langle e, j, \mathcal{B}_o\rangle$:
  - The type of obsel $e \in \mathcal{O}$.
  - The obsel's subject depends on the obsel's origin: the user, the environment and potential users or the robot.
  - The set of obsel related attributes $\mathcal{B}_o$. An attribute $y \in \mathcal{B}_o$ has a value $v^o : y \rightarrow \mathcal{D}_y \cup null$.

Example 3: The sequence of obsels in a trace for projecting video activity can start as following: robot: move to bedroom, user: yes, robot: propose film, user:no, robot: propose episode, user: yes .... An example attribute for the obsel "set volume" is the noise level in the environment.

From each trace, the system can extract several feedback rewards. The *feedback rewards* are gathered and processed later (through learning algorithms) to create the **Learned Rewards** that form reward function of the MDP decision model. Both *feedback rewards* and *learned rewards* have the same representation form.

When processing an activity trace, each time an obsel of type "Feedback" is encountered a feedback reward is created. The feedback reward holds in part the complete user profile and activity, both represented using their attribute values. However, as a result of learning algorithms, these attributes values can be generalised or described as *constraints* (logical expressions) over their possible values.

Definition 4: A feedback/learned reward is represented by a tuple: $\langle \mathcal{C}_p, \mathcal{C}_{ac}, id_{ac}, o_i, o_{i+1}, \mathcal{TR}, v\rangle$:

- $\mathcal{C}_p$: the constraints on the user profile attributes $\mathcal{B}_p$.
- $\mathcal{C}_{ac}$: the constraints on the activity attributes $\mathcal{B}_{ac}$.
- id_{ac}: the id of the activity.
- $o_i \in \mathcal{O}_r$: the robot action.
- $o_{i+1} \in \mathcal{O}_u$: the user feedback concerning robot action o_i.
- $\mathcal{TR}$: a backup of traces ids id_t that provoked the creation or a modification of this learned reward.
- $v = \mathcal{V}(o_{i+1})$ is the feedback value (the received reward). $\mathcal{V}$ is a predefined value function[2] that assigns a positive or negative weight for each feedback $\mathcal{V} : \mathcal{O}_u \rightarrow [-1, 1]$.

Example 4: For instance, a new feedback reward includes information about (1) the user profile attributes: $\mathcal{C}_p = (adult, male)$, (2) the activity $id_{ac} = video\_projection$ and activity related attributes $\mathcal{C}_{ac} = (evening, low\_brightness, low\_noise)$, (3) the robot's action $o_i = propose\_ferry\_cartoon$, (4) the user feedback $o_{i+1} = no$, (5) the id of the current analysed trace id_t, and (6) the value of the feedback $v = -1$. In a learned reward, constraints about user profile and/or activity attributes are represented like $\mathcal{C}_p = (adult, *)$ where $*$ represent any value for the attribute gender (male or female).

V. Robot Planning

The first question that we focus on for designing an adaptive companion robot is its ability to adapt and personalise its

[2]This function follows the assumption that the user's feedback is received directly after the robot action. It is possible to define a function $v = fct(\mathcal{V}(o_1), \ldots, \mathcal{V}(o_{i+1})$ using a heuristic (*e.g.* propagation).

behaviour according to the situation and its user, taking into account the user's profile. For this reason we chose to base the **Decision Planning** of the robot as a Markov Decision Process (MDP). An MDP is represented by a tuple $\langle S, A, T, R\rangle$ where: S is a set of states, A is the set of robot's actions, T is the transition function where $T(s, a, s')$ is the probability of transitioning from state s to state s' after doing action a and R is the reward function, where $r(s, a)$ is the robot's immediate reward for making action a while being in state s. An MDP policy π_{MDP} is a function $\pi : S \rightarrow A$, which associates an action to each MDP state. There are several algorithms to solve an MDP, classically Value Iteration [18] and Policy Iteration [19]. The complexity of such algorithms is $\mathcal{O}(|S^2||A|)$. If the agent has no knowledge about its environment, the transition and reward functions will be hard to calculate. In this case, the agent can directly learn its policy using reinforcement learning such as for Q-learning [20] that does not require prior knowledge about the transition function. Inverse reinforcement learning algorithms [21] extracts a reward function from an observed optimal behaviour (learn by observation). On the other hand shaping algorithms [6] extracts a reward function from human feedback on agent's behaviour.

In our problem, the reward function is not known for the robot, however, each user provide his feedback that represent his preferences on how the robot should behave. The algorithms that we propose in the next section create a reward function that is able to guide the robot through adapted and personalised behaviour.

VI. Learning From User Feedback

In this section, we present two **Learning From User Feedback** methods that we experimented through simulation. The first is a direct and certain algorithm, it can be considered as the optimal version for extracting a reward function that is certain at all times. The second generalises the learned rewards to be applied on unknown profiles and in unknown situations, this version is risky at its early stages of learning before it merges to a correct generalisation.

Both algorithms have mainly as entry a feedback reward fr extracted from an interaction trace and as output the set of learned rewards $\mathcal{LR}$ modified after processing fr. $\mathcal{LR}$ is then added as a part of the MDP reward function. The extraction of an $fr = \langle \mathcal{C}_p^{fr}, \mathcal{C}_{ac}^{fr}, id_{ac}^{fr}, o_i^{fr}, o_{i+1}^{fr}, \mathcal{TR}^{fr}, v^{fr}\rangle$ from a trace $\mathcal{T} = \langle id_p, id_{ac}, o_1, \ldots o_i, \ldots, o_n\rangle$ is done simply by filling the profile id_p and the activity id_{ac} attributes values in $\mathcal{C}_p^{fr}, \mathcal{C}_{ac}^{fr}$ respectively. o_i^{fr} is an obsel from $\mathcal{T}$ of type $\mathcal{O}_r$ that precedes another obsel $\mathcal{T}$ of type $\mathcal{O}_u$ (user feedback). The latter is set to be o_{i+1}^{fr}. $\mathcal{TR}$ holds the id of the trace $\mathcal{T}$ and v is the value of the user feedback $v = \mathcal{V}(o_{i+1})$.

A. The Certain Direct Learning Algorithm

The main idea of algorithm 1 is to add a feedback reward fr to the existing set of learned rewards lr in a simple and direct way. Several possibilities might occur that are presented in the algorithm.

1) If fr holds the same information as one $lr \in \mathcal{LR}$ (Lines 5:6); meaning that fr and lr have the same activity id id_{ac} and the same robot action o_i and that profile and activity attributes of fr satisfy their corresponding constraints in lr, then:
   a) If both fr and lr have the same feedback value direction (FD) (*i.e.* both v values in fr and lr are positive or negative (Line 7), then lr stays in $\mathcal{LR}$ after modifying the feedback value with $fct(v^{lr}, v^{fr})$ which can be a function that returns the average of both values (Line 8).
   b) If fr and lr have different feedback directions (FDs), then a *contradiction* is detected (the received feedback contradicts with the existing learned knowledge) and both rewards are flagged (added to $\mathcal{EC}$). The $\mathcal{EC}$ set is then sent to be reviewed by an expert in order to detect the reason of the contradiction (Lines 12:13). This contradiction might refer to missing attributes in the problem representation[3].
2) If fr holds a situation that is not included in any of the learned rewards, then fr is added as new learned reward in $\mathcal{LR}$ (Lines 14:15).

In addition to $\mathcal{LR}$, this algorithm has as output the set of contradicted rewards $\mathcal{EC}$ to be reviewed by the expert.

Algorithm 1 The direct learning algorithm

```
1:  INPUT fr = ⟨C_p^fr, C_ac^fr, id_ac^fr, o_i^fr, o_{i+1}^fr, TR^fr, v^fr⟩
2:  Output LR, EC
3:  Added = false, attention = false
4:  for all lr ∈ LR do
5:      if (id_ac^lr = id_ac^fr and o_i^lr = o_i^fr) then
6:          if (C_p^fr, C_ac^fr satisfy C_p^lr, C_ac^lr respectively) then
7:              if (v^lr ≥ 0 and v^fr ≥ 0) or (v^lr < 0 and v^fr < 0) then
8:                  Modify v^lr = fct(v^lr, v^fr).
9:                  Add the fr trace id to TR^l.
10:                 Added = true.
11:             else
12:                 Add fr and lr to EC.
13:                 Attention = true.
14: if (!added and !attention) then
15:     Add the reward fr to the set of learned rewards LR.
```

B. The Generalised with Risk Learning Algorithm

Algorithm 2 has a very different approach than Algorithm 1. We aim in this algorithm to be able to learn the reward function faster by minimizing the needed number of experiences. In addition, we aim to be able to generalise the learned function to be applied on unknown profiles and in unknown situations.

The main idea is to try to detect, for each possible robot action, the important set of attributes values (related to profile or activity) that affect the user feedback value direction (FD) (*i.e.* v is positive or negative). Attributes that are not important are generalised to any value (*) in all $\mathcal{LR}$ rewards (Line 17). This algorithm backs up all feedback rewards, so there is no loss of information because of the generalisation. The backup rewards are continuously used in the process of detection of

[3]A user might change his preference in a rainy day which informs us the need of adding weather forecast as an attribute.

important attributes. In the following, we describe, with examples, (a) how the algorithm uses contradictions between fr and generalised rewards in $\mathcal{LR}$ to detect important attributes and also (b) how it is possible to detect attributes that were falsely set as important in the first phase. Initially, and after receiving the first fr, the algorithm generalises all the attributes values to (*) with a reward value v equal to the value in fr.

Algorithm 2 The Generalised learning algorithm

1: **INPUT** $fr = \langle \mathcal{C}_p^{fr}, \mathcal{C}_{ac}^{fr}, id_{ac}^{fr}, o_i^{fr}, o_{i+1}^{fr}, v^{fr} \rangle$.
2: **Output** $\mathcal{LR}$, important attributes for each action.
3: Added = false.
4: **for all** $lr \in \mathcal{LR}$ **do**
5: **if** ($id_{ac}^{lr} = id_{ac}^{fr}$ and $o_i^{lr} = o_i^{fr}$) **then**
6: **if** ($v^{lr} \geq 0$ and $v^{fr} \geq 0$) or ($v^{lr} < 0$ and $v^{fr} < 0$) **then**
7: Added = true.
8: **else**
9: $\mathcal{BR}$ = related rewards to lr from backup rewards.
10: **for all** $br \in \mathcal{BR}$ **do**
11: Add br to $\mathcal{LR}$.
12: **for all** $att \in \mathcal{BR}$ **do**
13: **if** att is an important attribute **then**
14: Add att as important to action o_i^{fr}.
15: **if** (!added) **then**
16: Add the reward fr to the set of learned rewards LR.
17: Generalise all non important attributes in $\mathcal{LR}$.
18: Check for attributes that were falsely set as important.
19: Add fr to the backup set $\mathcal{BR}$.

a) Treating contradictions to extract important attributes: In Line 8 of Algorithm 2, a contradiction is detected between the fr feedback value and the lr feedback value. We will explain here, how the algorithm treats this contradiction to try to extract important attributes concerning the action o_i (Lines 9:14). First, a not empty set of backup feedback rewards that are related to lr is set to $\mathcal{BR}$. We mean by related that they concern the same robot action, they have the same FD and that the attributes values in br are included in the constraints of lr. In Figure 3 we show an example of how we use the backup rewards to detect an important attribute. In the example, we

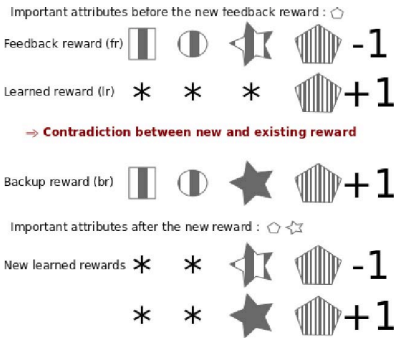

Fig. 3: Example of detecting an important attribute

notice the contradiction between fr and lr. We notice that fr respects the constraints in lr, however, they don't have the same FD. This contradiction leads to looking for the subset of backup rewards where each br respects all the constraints of lr and have the same FD. By comparing fr and br, we can conclude that, in addition to the "pentagon shape" attribute, the "star shape" attribute is also an important attribute that caused an opposite user reaction.

b) Checking for false important attributes: It is possible that the algorithm marks some attributes as important when they are actually not. Therefore, we added in Line 18 a function that checks for falsely detected important attribute. For example, if we have fr and lr who have the same FD with only one different attribute value, this attribute is marked as falsely detected important attributes and is removed from the list of important attributes for the concerned action if the following condition is true: there exist in the backup rewards a list of rewards covering all the combinations of *important* attributes with the same FD.

VII. Experiment Results

A. Experiment Details and Procedure

We experimented our representation and learning algorithms through simulation. In the simulation we used the activity of projecting a video to the user. The activity attributes and values corresponds to the examples presented in Section IV.

The procedure of evaluation is a loop of the following: (1) Re/Calculate the MDP policy. (2) Generate n traces. (3) Evaluate robots actions in the n traces. (4) Extract feedback rewards from traces, then learn and update the MDP reward function. (5) Repeat from step 1 until convergence.

The MDP represents in its states, the possible users' profiles (age and gender) and activity related information (noise, daytime, brightness and phase). The MDP set of actions represents all possible robot's actions (see Example 2). The transition function changes deterministically the phase value in the state knowing the robot action. A default reward function is given to respect the sequence of phases for realising the activity (the sequence is presented in Example 2). The MDP policy of the first loop of the evaluation respects only the sequence of phases without having any knowledge about users' preferences.

The generation of the traces (*i.e.* step 2) is made by simulation using the MDP policy for generating the robot action and some *predefined rules of preferences* to generate users feedback. We had predefined rules for each action of each phase which is connected to certain attributes of the activity or the profile (*e.x.* the room selection for the projecting example depends on the age of the user and the day time, and the video type depends on the age and the gender).

We evaluated both algorithms on the same randomly generated situations (user profile and activity related information). We followed the procedure described earlier with $n = 100$ traces. For the traces evaluation (*i.e.* step 3), after each n traces we calculated the number of robot's actions that were followed with a negative user feedback (negative actions). We also compared the number of complete positive traces, where the robot succeeded to achieve a complete activity with the user without receiving any negative feedback.

B. Results

Results shown in Figure 4 present the convergence of both learning algorithms to an optimal adaptive and personalised

behaviour with no negative actions. It shows that the second algorithm with generalisation creates 10% of negative actions during the first 100 traces and converges to 0% negative actions after 500 with exception. In the first 100 traces, there was 66 completely positive experiences resulting from the algorithm with generalisation. We confirm that this algorithm was able to learn the dependencies between the profile and activity attributes with each robot action. Important attributes that were predefined for the simulation were completely learned at the end of the experiment. On the other hand, the direct algorithm started with 37% of negative actions during the first 100 traces and converged after 1000 traces. The first 100 traces included 42 positive experiences only.

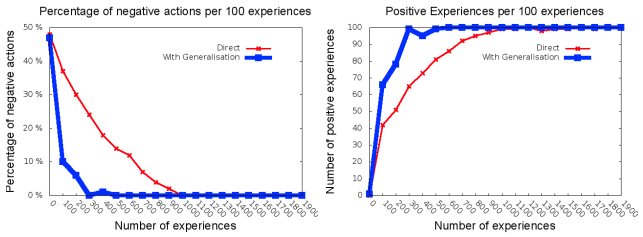

Fig. 4: Comparison between the direct algorithm and the algorithm with generalisation: the number of negative actions and the number of positive experiences (with zero negative feedback) presented per 100 experience

In the evaluated simulations, the predefined rules of preference were followed at all times. This means there was no exceptions in users' preferences. However, we know that this is far from true in real applications. For example, most but definitely not all male adults like to watch sport related programs. For this reason, we are currently working on handling exceptions and holding probabilities on rules of preferences. For personalised services, the robot must link exception preferences to the exact user.

VIII. Conclusion and Future Work

In this paper, we presented an adaptive companion robot decision model that learns users' preferences using their feedback. Two algorithms were proposed, the first is simple and certain while the second generalised knowledge with risk. The generalised algorithm can learn the dependencies between each robot action and attributes specifying user profile and/or activity. It converges to an optimal adaptive and personalised reward function with less than half experiences needed by the direct algorithm. These results show that the proposed architecture and learning algorithms permit the companion robot to adapt to its users' preferences and even adapt to first time users knowing only basic information about their profiles.

We are actually working on evaluating our results with the EMOX robot. In future work, we would like to work on a higher level of adaptation, where the robot can adapt during activity selection. Also, we would like to work on updating/completing users' profiles by analysing the interaction traces (*e.g.* detecting personal preferences like alarm time).

References

[1] M. Saerbeck and T. Schut, "Expressive robots in education: varying the degree of social supportive behavior of a robotic tutor," *In CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pp. 1613–1622, 2010.

[2] C. Breazeal, J. Gray, and M. Berlin, "An Embodied Cognition Approach to Mindreading Skills for Socially Intelligent Robots," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 656–680, May 2009.

[3] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards robotic assistants in nursing homes: Challenges and results," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 271–281, Mar. 2003.

[4] T. Kanda and H. Ishiguro, "Communication robots for elementary schools," *Proceedings of AISB'05 Symposium Robot Companions: Hard Problems and Open Challenges in Robot-Human Interaction (Hatfield Hertfordshire)*, pp. 54–63, 2005.

[5] A. Tapus, C. Tapus, and M. J. Matarić, "User-robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy," *Internationl Journal on Intelligent Service Robotics*, vol. 1, no. 2, pp. 169–183, Feb. 2008.

[6] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: the tamer framework," in *Proceedings of the fifth international conference on Knowledge capture*, ser. K-CAP '09. New York, NY, USA: ACM, 2009, pp. 9–16.

[7] M. Pollack, L. Brown, and D. Colbry, "Pearl: A mobile robotic assistant for the elderly," *Workshop on Automation as Caregiver: the Role of Intelligent Technology in Elder Care (AAAI)*, 2002.

[8] A. B. Karami and A.-i. Mouaddib, "A Decision Model of Adaptive Interaction Selection for a Robot Companion," *proceedings of the 5th European Conference on Mobile Robots, ECMR'11*, pp. 83–88, 2011.

[9] A.-B. Karami, L. Jeanpierre, and A.-I. Mouaddib, "Partially Observable Markov Decision Process for Managing Robot Collaboration with Human," *2009 21st IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 518–521, Nov. 2009.

[10] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita, "Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 218–225, 2005.

[11] D. Clauzel, K. Sehaba, and Y. Prié, "Enhancing synchronous collaboration by using interactive visualisation of modelled traces," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 84–97, jan 2011.

[12] A. Kobsa, "User modeling and user-adapted interaction," in *CHI Conference Companion*, C. Plaisant, Ed. ACM, 1994, pp. 415–416.

[13] B. Encelle and N. Baptiste-Jessel, "Personalization of user interfaces for browsing xml content using transformations built on end-user requirements," in *W4A*, ser. ACM International Conference Proceeding Series, S. Harper and Y. Yesilada, Eds., vol. 225. ACM, 2007, pp. 58–64.

[14] A. M. Hussaan and K. Sehaba, "Adaptive Serious Game for Rehabilitation of persons with cognitive disabilities," in *The 13th IEEE International Conference on Advanced Learning Technologies*, Jul. 2013, pp. 1–5.

[15] N. P. Cuntoor, B. Yegnanarayana, and R. Chellappa, "Interpretation of state sequences in hmm for activity representation," *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP).*, Jan. 2005.

[16] M. Albanese, R. Chellappa, N. P. Cuntoor, V. Moscato, A. Picariello, V. S. Subrahmanian, and O. Udrea, "A constrained probabilistic petri net framework for human activity detection in video," *IEEE Transactions on Multimedia*, vol. 10, no. 6, pp. 982–996, 2008.

[17] J. Muncaster and Y. Ma, "Activity recognition using dynamic bayesian networks with automatic state selection," in *Proceedings of the IEEE Workshop on Motion and Video Computing*, ser. WMVC '07. Washington, DC, USA: IEEE Computer Society, 2007, p. 30.

[18] R. Bellman, "A Markovian Decision Process," *Indiana University Math. J.*, vol. 6, pp. 679–684, 1957.

[19] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.

[20] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press/Bradford Books, 1998.

[21] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 663–670.