Adaptive Artificial Companions learning from users’ feedback

Abir B. Karami∗ and Karim Sehaba† and Benoit Encelle‡

Abstract
Until recently, propositions on the subject of intelligent service companions, like robots, were mostly user and environment independent. Our work is part of the FUI-RoboPopuli project, which concentrates on endowing entertainment companion robots with adaptive and social behavior. More precisely, we focus on the capacity of an intelligent system to learn how to personalize and adapt its behavior/actions according to its interaction situation that describes (a) the current user(s) attributes, (b) the current environment attributes. Our approach is based on models of the type of Markov Decision Processes (MDPs) that are largely used for adaptive robot applications. In order to have, as quickly as possible, a relevant adaptive behavior whatever the interaction situation, several approaches were proposed to decrease the sample complexity required to learn the MDP model, including its reward function. We argue that systems that are based on detecting important attributes for each decision are more likely to converge faster than others. To this end, we present two algorithms to learn the MDP reward function through analyzing interaction traces (i.e. the interaction history between the robot and its users including their feedback regarding the robot actions). The first algorithm is direct, certain and does not particularly exploit its knowledge to adapt to unknown situations (i.e. unknown users and/or environment settings). The second is able to detect the importance of certain situation attributes in the adaptation process. The detection of important attributes is used to speed up the learning process by generalizing the learned reward function to unknown situations. In this paper, we present both learning algorithms, simulated experiments and an experiment with the EMOX robot that was upgraded during the FUI-RoboPopuli project. The results of those experiments prove that when dealing with adaptive decision making, the detection of important attributes for each decision speeds up the learning process and help in achieving convergence using much less number of samples. We also present a scaling analysis through the simulated experiments.

Keywords
Adaptive behavior; Personalization; Learning from users’ feedback; Interaction Traces; Markov Decision Processes (MDPs); Companion Robots.

∗Université Lille Nord de France, École des Mines de Douai, F-59500 Douai, France
†University of Lyon 2, LIRIS, UMR CNRS 5205, F-69676, Lyon, France
‡University of Lyon 1, LIRIS, UMR CNRS 5205, F-69622, Lyon, France
1. Introduction

Recent studies concentrate on user dependent approaches for companion robots applications. Whether in educational (Saerbeck and Schut (2010); Howley et al. (2014)), social and nursing (Breazeal et al. (2009); Kachouie et al. (2014); Silva and Romero (2012)), or service applications (Breuer et al. (2012)), it is expected that robots take into account the fact that their environment includes several kinds of users with different age, preferences, needs, etc. Thus, more interest has been oriented towards user dependent approaches.

Recent work on personalized service robots are mostly based on predefined rules of adaptation. In Kanda and Ishiguro (2005), a companion robot for children proposes interactions by following a predefined order of execution depending on the history of interaction with the child (e.g. if not yet presented, the robot presents itself to the child before singing a song with him). In Mason and Lopes (2011), the authors propose a personalized system that learns through interactions. They represent the environment as a vector in an attribute space. The user preference is defined as a preferred goal state (vector), that the robot learns to achieve by following the user verbal instructions. After several interactions, the robot learns to classify a state as desired or not. The authors created a large number of attributes and used the results of an online survey to select the relevant ones for a particular task. The relevant attributes were not learned automatically. Furthermore, it is interesting if personalized service robots have the ability to adapt by connecting the most suitable behavior to some user profile and environment particularities (situation attributes). This would help in generalizing the learned behavior to other, possibly unknown, users. This is the main motivation of our proposition in this paper.

Human feedback is used differently, by diverse approaches, for planning processes. A human feedback can be considered as a source of information. In Rosenthal et al. (2011a,b), the authors study the availability of environment occupants to help the robot. Other approaches use the human feedback as shaping signals to learn how to achieve a task. In such approaches, the source of the feedback is considered as an observer of the system who evaluates each of the robot’s actions (Knox and Stone (2009); Knox et al. (2013)) or the robot entire policy (Akrour et al. (2012)). The TAMER framework (Knox and Stone (2009)) proposes a method to shape a learning robot by giving positive and negative signals (as for a domestic dog). This method helps in training the robot to execute a task. These approaches are not proposed to learn personal preferences nor to be used in multi-user environments.

In this article, we focus on designing an adaptive system for a companion robot that is able to adapt and personalize its behavior according to the current situation. We focus in particular on two questions:

1. How the robot can adapt according to the interaction situation that includes a user profile (ex. age, preferences, habits, …) and the environment settings (time, place, brightness level, noise level, …)? For this problem, our approach is based on Markov Decision Process models (MDP) (Bellman (1957)) where all needed pieces of information about the user profile and the activity are integrated in the MDP state. A correct MDP reward function is needed to generate an adapted and personalized robot behavior 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 is based on the analysis of the history of interactions between the robot and the users in order to determine the reward function. The interaction traces include a sequence of robot actions, some activity related environmental settings (i.e. brightness level, noise level) user feedback regarding the robot’s actions. Each feedback is analyzed by the robot to learn more about the users’ preferences and update the reward function. Our approach also generalizes the rewards by learning the dependence between the robot behavior and certain information about the user and/or the activity. This helps the robot to better adapt to new users and to new situations.

In the following: we present in Section 2 the related work. Section 3 shows the context of our research with an illustrative scenario. We present the proposed general architecture and knowledge representation model for an adaptive and personalized robot system in Section 4. In Section 5, we briefly present the approach we use for the robot planning system based on an MDP model. We present, in Section 6,
our proposed learning algorithms to learn the reward function that will help the robot planner in its adaptation process. We then present our experimental results. In Section 7, we present our first experiment to evaluate, by simulation, the robot performance using our proposed learning algorithms. We also present in this section convergence and scaling analysis of the system. The second experiment, presented in Section 8, is with a real robot showing the applicability of our method on a real scenario of a companion robot and the acceptability of the robot behavior by the users. Finally, we conclude and present our future work.

2. Related Work

Most approaches for adaptive robots use models based on probability theory and statistics, like Markov decision models. For example, the approach proposed for Pearl-Nursebot is based on Partially Observable Markov Decision Process (POMDP) (Pollack et al. (2002); Pineau et al. (2003)). A POMDP is a model to calculate optimal measures of control in uncertain environments. In order to decrease the complexity of solving the POMDP, the authors proposed a hierarchical solution 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 a robot that adapts its decisions based on the estimated need of the human partner (Karami and Mouaddib (2011); Karami et al. (2009)). The robot observes human actions, calculates a probability of his possible intentions and chooses the appropriate interaction type (assistance, cooperation, collaboration). To overcome the complexity of solving the POMDP, they proposed a solution based on dividing the intention estimation problem from the decision making by using a Hidden Markov Model (HMM) and an MDP. In these approaches, the robot adapts regarding the situation and the inferred need of a user, however, it does not propose personalized behavior according to a user profile nor consider a user feedback.

Other approaches aim to learn actions that maximize certain interaction results using reinforcement learning methods. A system is proposed for an assistant robot through rehabilitation exercises (Tapus et al. (2008)). This system learns to adapt its behavior and personality through three main parameters: distance of interaction, movement speed and verbal communication (volume and speed of speech). The authors use a Policy Gradient Reinforcement Learning (PGRL) algorithm to optimize these three parameters according to the personality of the user in order to maximize his performance (number of exercises on a certain period of time). A similar approach, that also uses PGRL, aims to observe the subconscious reactions of discomfort from a human and adapt the distance of interaction, the gaze and the movement speed aiming to minimize these reactions and maximize the comfort of the user (Mitsunaga et al. (2005)). In these approaches, users feedback (performance or reactions) is used in the process of learning. However, the exact information about the users that is related to the robot decision (e.g. personality, discomfort reactions) is specified in advance of the learning process and is not detected or learned automatically.

Some approaches analyze the interaction traces to adapt to users. In Sehaba (2012), the author proposes an approach to allow learners with different skills, abilities or preferences to exchange, among themselves, the traces of their own activities. Through transformation processes, the shared traces are adapted according to the profile of its target user. Another system is proposed for “Robovie”; the school companion for children (Kanda and Ishiguro (2005)). The robot can identify the students and memorize the history of interaction with each of them. Therefore, it is able to personalize its interactions toward 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 and the adaptation knowledge are predefined as rules and not learned automatically.

Several approaches use human observer feedback to learn a certain behavior (Vidal et al. (2013); Knox et al. (2013)). In Knox and Stone (2009), the authors propose 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 sends a feedback signal as judgement on the action quality. This approach uses supervised learning techniques to model a person’s reinforcement function and then uses the learned model to choose actions that should maximize the reinforcement. It is used to teach an agent how to achieve certain tasks. However, it is not proposed for interactive environments where the agent should adapt to its current user nor to handle environments including several users with different preferences. Other related approaches use semi-supervised learning, like active learning methods, where agents interactively query the user feedback on certain situations that are estimated important in the learning process. However, such mechanisms are difficult to use in companion robot scenarios where the system is not in control of current situations nor the current user. Classic supervised learning techniques are also not appropriate to use for problems where feedback are received from users (not experts) and where the model to be learned is not necessarily determined (users preferences can change over time).

For personalized robot behavior in multi-user applications, it is difficult to manually predict the adaptive knowledge for all possible situations. Furthermore, in certain scenarios, it is practically impossible for a robot designer to define the needed knowledge (adaptive knowledge) that covers all possible user profiles and suits to all possible situations. For this reason, we argue that users preferences (represented in a reward function) can be learned through analyzing the history of interactions. The proposed approach works under the assumption that users can provide positive and negative feedback regarding robot actions. The user feedback can be collected from facial expressions, gestures, vocal expressions, etc. In addition, we argue that approaches that detect the relevance of each situation attribute (environment and user profile attributes) in each type of decision learn faster than those that does not detect such relevance. Such knowledge helps in generalizing the reward function and allows the robot to adapt to new situations (e.g. with new users), and more importantly helps in decreasing the complexity of convergence to an optimal adaptive policy.

3. Illustrative Scenario of an Adaptive Robot

We present in this section a general example that will serve as an illustrative example throughout the following sections.

In this work, we are interested in endowing a companion robot (Figure 1a) with adaptive and social behavior. Such robot can propose entertainment activities to house occupants. Let us consider the activity of projecting a video for a user (Figure 1b). 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 information that we will call attributes of the activity should be set by the robot. The robot decisions should respect the interaction situation (the environmental settings and the user preferences) in addition to some general rules (e.g. rules set by the parents for their children). A user preference can be part of a general rule (e.g. male adults like to watch sport) or it can be personal (e.g. user X needs a higher level of volume because of an audition problem).

![EMOX](a) EMOX: A turtle bot composed of a camera, laser range-finder, pico projector, microphone and speaker.

![EMOX](b) An older version of EMOX, projecting a cartoon movie on the wall.

**Fig. 1.** EMOX (EMOtion eXchange) platform, the augmented robot of Awabot Corporation.
Users preferences regarding the activity are not necessarily known by the robot. We are interested in developing an approach to help learning these preferences by analyzing users’ profiles and the interaction traces including users’ feedback. The source of feedback can be different e.g. facial expressions, gestures, vocal expressions, etc. In our experiments, we only used discrete positive and negative signals as satisfaction feedback from users (-1, 0, 1), however, more complicated visual and vocal expression quantifiers can be added to the system. The robot learns the adaptive rules (reward function) and uses them to personalize its decisions.

We notice from this example that each robot decision can be affected by certain activity attributes. For example, the robot decision about what kind of video to project can be affected by some user profile attributes (e.g. user gender or age) and maybe also by the time of the day or the place of interaction, etc. On the other hand, the decision about the level of brightness is barely related to the user gender. In Section 6 we propose an algorithm that is able to detect the set of important attributes for each kind of decision and we explain how the learned reward function can be generalized in consequence to cover unknown situations. Then, we analyze as part of our experiments the effects of such generalization on accelerating the learning process.

4. General Architecture and Knowledge Representation

In this section, we describe the general architecture of the robot system represented in Figure 2. An earlier version of this architecture is detailed in Karami et al. (2013b). The system input represents observations of the user behavior, including his/her feedback, and environment related observations, including observations of potential users. We refer by user, the person who is sharing the activity with the robot and potential users, those who have-been/might-be in the past/future sharing an activity with the robot. The system output represents the robot’s actions. As shown in Figure 2, the system includes 4 knowledge bases: interaction traces, activities, users profiles and learned rewards (detailed in this section) and two main processes: decision process and learning from feedback process (described in Section 5 and 6 respectively). In the following of this section we will present formal descriptions of an activity, a user profile, an interaction trace and a learned reward. We will use, for clarification purpose, some examples inspired from the illustrative scenario described in Section 3.

The activities database holds a description of all possible activities that the robot can share with or propose to its users (project a video, play radio, …). For each activity, we denote \( B_{ac} \), the environmental attributes where \( B_{ac} = \langle b_{ac0}, b_{ac1}, b_{ac2}, \ldots b_{acn} \rangle \). Each attribute \( b_{aci} : i \in \{0, n\} \) has its domain of values \( D_{bac} \).

Example 1. In reference to our illustrative example, the list of attributes of the activity of projecting a video to the user (\( B_{video} \)) might include the level of noise in the environment (\( b_{video} = noise \)), where \( D_{video_{noise}} = \{low, medium, high\} \).

Users profiles hold information about users. Each user is represented by a profile \( P \). A profile contains all pieces of information concerning the user that might be useful in the adaptive decision process. These pieces of information are represented as attributes \( B_p = \langle x_0, x_1, \ldots x_m \rangle \). Each attribute \( x_i : i \in \{0, m\} \) has its domain of values \( D_{x_i} \). On the contrary of activities attributes where each activity might have a different set of attributes, all profiles are described with the same set of attributes.

Example 2. In addition to the profile id, a profile can include attributes like \( x_1 = age \) with \( D_{age} = \{child, teenage, adult\} \); and \( x_2 = gender \) with \( D_{gender} = \{female, male\} \).

We based our model of interaction traces on the model defined by Clauzel et al. (2011). Generally speaking, a trace
is a set of obsels (OBServed ElemenT(S) ordered by the time of their occurrence. Obsels are generated during interactions between the system (e.g. the robot) and the environment (including the users). Formally, the obsel has an origin (e.g. the user, the robot or the environment), and a set of attributes that characterizes the observed event.

We denote input obsels related to the user \( O_u \) and the environment \( O_e \). Both \( O_u \) and \( O_e \) are lists of unquantified observations. The robot actions are integrated in the trace as output obsels \( O_r \). We note the set of all obsels: \( O = O_u \cup O_e \cup O_r \). As part of user related obsels \( O_u \) is his feedback which will be analyzed to learn users preferences and how to adapt to them.

In our model, a Trace includes the sequence of input/output during one activity between the robot and a user. It is represented as described in Definition 1.

**Definition 1.** A trace \( T \) created during an activity ac is represented by a tuple :
\[
\langle id_t, id_p, id_{ac}, o_1, \ldots, o_n \rangle \text{ where,}
\]
- \( \text{id}_t, \text{id}_p, \text{and } \text{id}_{ac} \) respectively the trace id, the user id and the activity id,
- \( o_1, \ldots, o_n \): the sequence of obsels representing the traced activity, where \( o_i \in \langle e, j, B_o \rangle; \)
  - The type of obsel \( e \in O \),
  - The obsel’s origin \( j \): the user \( u \), the environment and potential users \( e \) or the robot \( r \),
  - The set of obsel related attributes \( B_o \). An attribute \( y \in B_o \) has a value \( v^o(y) : B_o \rightarrow D_y \cup \text{null} \).

**Example 3.** The sequence of obsels in a trace for projecting video activity can start as following: \( O_r \): move to bedroom, \( O_u \): yes, \( O_r \): propose film, \( O_u \): no, \( O_r \): propose episode, \( O_u \): yes ... An example attribute for the obsel “move to bedroom” is the actual position of the robot in the environment.

From each trace, the system can extract several feedback rewards. When processing a trace, each encountered obsel of type robot action and its associated feedback (if exists) (see Assumption 1) are used to create a feedback reward.

**Definition 2.** A feedback (learned) reward is represented by a tuple: \( \langle \mathcal{C}_p, \mathcal{C}_{ac}, \text{id}_{ac}, o_1, o_{i+1}, TR, v \rangle \), where:
- \( \mathcal{C}_p \): the constraints on the user profile attributes \( B_p \),
- \( \mathcal{C}_{ac} \): the constraints on the activity attributes \( B_{ac} \),
- \( \text{id}_{ac} \): the id of the activity,
- \( o_1 \in O_r \): the robot action,
- \( o_{i+1} \in O_u \): the user feedback concerning robot action \( o_i \),
- \( TR \): a backup of traces ids that provoked the creation or a modification of this learned reward. In a feedback reward, \( TR \) holds only the id of the original trace that generated it,
- \( v = V(o_{i+1}) \): a quantified value of the feedback \( o_{i+1} \) (the received reward) over the robot action \( o_i \). \( V \) is a predefined value function that assigns a positive, negative or null weight for each feedback \( V : O_u \rightarrow [-1, 1] \).

**Example 4.** For instance, a new feedback reward includes information about (1) the user profile attributes: \( \mathcal{C}_p = (\text{adult}, \text{male}) \), (2) the activity \( \mathcal{id}_{ac} = \text{video and activity attributes } B_{\text{video}} = (\text{evening, low_brightness, low_noise}) \), (3) the robot’s action \( o_i = \text{propose_fairy_cartoon} \), (4) the user feedback \( o_{i+1} = \text{no} \), and (5) the value of the feedback \( v = V(\text{no}) = -1 \). In a learned reward, constraints about user profile and/or activity attributes are represented like \( \mathcal{C}_p = (\text{adult, *}) \) where * represent any value from gender attribute domain \( D_{\text{gender}} \) (defined in Example 2).

The extraction of feedback rewards from a trace is presented under the following assumption:

**Assumption 1.** A user feedback is received after the robot action and is quantified using the function \( V \) (seen in Definition 2). A lack of feedback for an action can be quantified as positive, negative or null value depending on the scenario or learned user habits. Considering the problem of the interval between the robot action and the user reaction, it is possible to define a function \( V(o_{i+1}) = h(V(o_1), \ldots, V(o_{i+1})) \) using a heuristic (e.g. propagation) (Hockley (1984); Knox and Stone (2009)).

Each trace can generate one or more Feedback reward. Every component of a feedback reward \( fr = \langle \mathcal{C}_p, \mathcal{C}_{ac}, \text{id}_{ac}, o_1, o_{i+1}, TR, v \rangle \) (Definition 2) is filled from a trace \( T = \langle id_t, id_p, id_{ac}, o_1, \ldots, o_n \rangle \) (Definition 1) as follows:
- the corresponding profile and activity attributes values are filled in \( \mathcal{C}_p, \mathcal{C}_{ac} \) respectively,
• \( o_{i+1} \) is an obsel from \( T \) of type \( O_r \),
• \( o_{i+1}^{fr} \) is the obsel of type \( O_u \) that proceeds \( o_{i}^{fr} \) in \( T \),
• \( TR \) holds the id of the trace \( T \),
• and \( v \) is the quantified value of the user feedback \( v = V(o_{i+1}) \).

The feedback rewards are then processed (through learning algorithms) to create the learned rewards that are integrated in the reward function. Both feedback rewards and learned rewards have the same representation form (Definition 2). The feedback reward holds in part the complete user profile and activity attributes values. However, during the learning phase, these attributes values can be generalized or described as constraints (logical expressions) over their possible values in a learned reward.

### 5. Decision Process

The system can use any decision process that allows the integration and the use of the learned rewards (adaptation knowledge). For simplicity and generality, we chose to use a classic Markov Decision Process (MDP) in our experiments. However, other decision processes might be more appropriate depending on the framework properties, like for example Partially Observable Markov Decision Processes (POMDPs) (Kaelbling et al. (1998)) for partially observable environments or contextual multi-armed bandit (Loftin et al. (2014)) for non-sequential decision processes.

Formally, an MDP is represented by a tuple \( \langle S, A, T, R \rangle \) where: \( S \) is a finite set of states; \( A \) is a finite set of agent’s actions; \( T \) is a state transition function with \( T(s, a, s') = Pr(s_t = s_t|s_{t-1} = s, a_{t-1} = a) \) representing the probability of transitioning from state \( s \) to state \( s' \) after doing action \( a \), and \( \sum_{s' \in S} T(s, a, s') = 1 \forall (s, a) \); and \( R \) is a reward function mapping \( S \times A \times S \) to a real number that represent the agent’s immediate reward for making action \( a \) while being in state \( s \) and ending in state \( s' \). The objective of the agent is to calculate a policy \( \pi : S \rightarrow A \) which assigns for each possible state an optimal action that maximizes the long-term expected reward \( E[\sum_{t=0}^{\infty} \gamma^t r_t] \) where \( \gamma \) is a discount factor and \( r_t \) is the reward at time \( t \). There are several algorithms to solve an MDP, classically Value Iteration (Bellman (1957)) and Policy Iteration (Howard (1960)). The complexity of such algorithms is \( O(|S|^2||A|) \).

We use an MDP to plan the robot decisions. The desired robot behavior is adaptive according to the situation (thus personalized to the user). We will detail the way we define the MDP model in the following example.

**Example 5.** In this example we will present, for simplicity, the MDP model for the activity of projecting a video. Such activity can be achieved in phases. For example, starting by moving to the right room (e.g. living room, children bedroom, etc.), then choosing the length of the video (e.g. film or episode), then choosing the type of the video (e.g. fairy cartoon, science fiction, drama, etc.), then setting the sound level (e.g. low, medium or high), then setting the brightness level (e.g. low, medium or high) and finally, start projecting the video. For this activity, the set of states \( S \) represents the user profile attributes (e.g. gender, age, etc.), the activity attributes (e.g. level of brightness, level of noise, etc.) and the actual phase of the activity. The set of actions \( A \) represents all possible robot actions (e.g. propose a video, set the brightness level, set the volume level, etc.). 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 \( A \) (e.g. the action of setting the volume will change the volume attribute value and if the user is satisfied the system will transition to a state representing the next phase).

In our experiments the transition function is known and deterministic. The initial reward function is defined in a way that guides the system through a behavior that only respects the sequence of phases. However, through interaction this function is enriched in order to guide the system through more adaptive and personalized behavior according to learned preferences.

### 6. Learning From User Feedback

The learning process allows the robot to learn the preferences and habits of users by analyzing the interaction traces. We present, in the following, a direct and certain algorithm that can be considered as a safe version for building a certain at all times reward function. This algorithm is presented for the sake of comparison. We mainly propose in this section a learning algorithm that generalizes the learned rewards to be applied on unknown situations (unknown profiles or environmental settings). The algorithm with generalization
should permit the system to converge with much less number of interactions by detecting the important attributes for each decision. The main drawback of this algorithm is the fact that it is with risk until it converges to a correct generalization. The gain in number of needed interactions and the risk in performance before converging are evaluated in our experiments in Section 7.

Both algorithms learn from an input set of Feedback Rewards ($\mathcal{FR}$). Feedback rewards are extracted from the interaction traces. The output of both algorithms is the set of learned rewards $\mathcal{LR}$ modified after processing every feedback reward $fr$. $\mathcal{LR}$ is then added to the MDP reward function.

6.1. The Certain Direct Learning Algorithm

In this algorithm (Algorithm 1), the addition of the feedback reward to the learned rewards $\mathcal{LR}$ is done in a simple and direct way:

1. If $fr$ holds the same information as one $lr \in \mathcal{LR}$; 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 $lr$ stays in $\mathcal{LR}$ after modifying the feedback value with $fct(v^{fr}, v^{lr})^2$.
   - In the case where $fr$ 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 can be further analyzed with the concerned user (and/or the system designer) to detect the reason of the contradiction: an error regarding the user feedback or missing attributes in the problem representation.2

2. If $fr$ holds a new situation and does not match to any of the existing $lr \in \mathcal{LR}$, then $fr$ is added as new learned reward in $\mathcal{LR}$.

This means that for each received $fr$ there can be at most one $lr \in \mathcal{LR}$ that matches it and in this case lines (7:11) are applied. In case no match exists, $fr$ is added as a new $lr$ (line 13).

6.2. The Generalized Learning Algorithm

We aim in this algorithm (Algorithm 2) to learn the reward function faster by minimizing the needed number of experiences. In addition, we aim to be able to generalize the learned function to generate adaptive behavior in unknown situations (new user profiles and/or new environmental settings). The used mechanism in this algorithm is inspired from the version space generalizing and specializing techniques. The version space (Russell and Norvig (2003)) is a machine learning approach used for binary classification. Its major drawback is its inability to deal with noise, which means that any detected contradiction can cause the version space to fail in the learning process.

Differently from the first algorithm, this one tries to detect important attributes for each possible robot action $o_i$ during an activity $ac$. We denote $\mathcal{IA}$ the set of important attributes concerning the action $o_i$. An important attribute (related to a profile or the environment) is one that its value affects the user-Feedback-value-Direction (FD) (i.e. $v$ is positive or negative). Attributes which their values are not important are generalized to any value (*) in all $\mathcal{LR}$ rewards of the concerned action $o_i$ and activity $ac$ (line 5 of Algorithm 2).

This algorithm backs up all feedback rewards (line 23), so there is no loss of information because of the generalization. $\mathcal{FR}$ represents the set of all treated feedback rewards (the backup set). The backup rewards $\mathcal{FR}$ are continuously

---

2 In our experiments, we used the average of both feedback values $fct(v^{fr}, v^{lr}) = average(v^{fr}, v^{lr})$.

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

---

Algorithm 1 The direct learning algorithm

1: INPUT $fr = (C^{fr}_p, C^{fr}_{ac}, id_{ac}, o_i^{fr}, v^{fr}, T^{fr}, \mathcal{LR})$
2: Output $\mathcal{LR}$, $\mathcal{EC}$
3: Added = false
4: for all $lr \in \mathcal{LR}$ do
5:  if $(id_{ac} = id_{ac}^0$ and $o_i^r = o_i^{fr})$ then
6:    if $(C^{fr}_p, C^{fr}_{ac}$ satisfy $C^{fr}_p, C^{fr}_{ac}$ respectively) then
7:       Modify $v^{lr} = fct(v^{fr}, v^{lr})$.
8:      Added = true.
9:      if $(v^{fr} \geq 0$ and $v^{lr} < 0$ or $v^{fr} < 0$ and $v^{lr} \geq 0$) then
10:     Add $fr$ and $lr$ to $\mathcal{EC}$.
11:   if (added) then
12:     Add the reward $fr$ to the set of learned rewards $\mathcal{LR}$.
used in the process of detecting important attributes. In the following, we describe, using the example presented in Figure 3, (a) how the algorithm uses contradictions between newly received $fr$ and the generalized rewards in $LR$ to detect important attributes and also (b) how it is possible to detect attributes that were falsely set as important in the first phase.

**Algorithm 2** The Generalized learning algorithm

1. **INPUT** $fr = (c^fr, c^fr, id^fr, o^fr, o^fr, v^fr), LR, IA, FR$.
2. **Output** $LR, IA, FR$.
3. Added = false.
4. **for all** $lr \in LR$ do
5. if $(id^fr, id^fr, and o^fr = o^fr)$ then
6. if $(c^fr, c^fr$ satisfy $c^lr, c^lr$ respectively) then
7. if $(v^fr \geq 0 and v^fr \geq 0 or v^fr < 0 and v^fr < 0)$ then
8. Modify $v^lr = fct(v^fr, v^fr)$.
9. Added = true.
10. **else**
11. Added = false.
12. $RR =$ related rewards to $lr$ from $FR$.
13. **for all** $rr \in RR$ do
14. Add $rr$ to $LR$.
15. **for all** $att \in rr$ do
16. if $(att \notin IA_o, and att^rr \neq att^fr)$ then
17. Add $att$ to $IA_o$.
18. **if** (Added) then
19. Add the reward $fr$ to the set of learned rewards $LR$.
20. Generalize all non important attributes in $LR$.
21. Confirm detected important attributes are important.
22. Check $LR$ for redundancies and contradictions.
23. Add $fr$ to the backup set $FR$.

Figure 3 presents 3 time-ordered feedback rewards (left side) concerning the same activity and robot action. These feedback rewards are processed in the received order generating the set of learned reward (right side) which is updated at each phase of the process. In the figure, activity and profile attributes are represented in the set $AT$. The set of attributes and their possible values are described in the figure (corner left). The constraints over a profile or an activity attribute (line 6 of Algorithm 2) is considered satisfied if the attribute value is similar in both $fr$ and $lr$ or if the $lr$ attribute value is (*) . The feedback and learned rewards are represented in a simplified manner in Figure 3. Knowing that all represented rewards concern the same robot action and activity, those are not shown in the example. Each reward represents the exact value of each attribute and the value of the feedback reward $(v_{at1}, v_{at2}, v_{at3}) = v(o_{a+1})$. In the example, each new feedback reward produces a contradiction with, what is considered at the time of processing, the current set of $LR$.

- **Treating contradictions to extract important attributes:**
  - The algorithm searches for a contradiction between the feedback values of the received $fr$ and one $lr$ from the $LR$ set of learned rewards (line 10). If a contradiction is detected, the algorithm tests if there is an important attribute concerning the action $o_i$ and the activity $ac$ (the shared robot action and activity in both $fr$ and $lr$) (lines 12:17). First, a not empty set of backup feedback rewards that are related to $fr$ is set to $RR$ (line 12). We consider an $rr \in RR$ is related to $lr$ if they both concern the same robot action and activity, they have the same FD and that the attributes values in $rr$ are included in the constraints of $lr$ (or equal to the attributes of $lr$). Second, the algorithm looks for attributes in $rr$ with different values than $fr$ (line 16). If found, the concerned attributes are added to the set of important attributes for the action $o_i (IA_{o_i})$ (line 17).

In the example shown in Figure 3, after receiving the second $fr$, a contradiction is detected between the latter and the existing $lr$. As shown in red in the figure, two important attributes are detected (the circle and the square). After any update in the set of important attributes, the algorithm updates the set of learned rewards by generalizing all attributes that are not set as important by setting their values to (*) (line 20).
- Checking for false important attributes:
It is possible that the algorithm marks some attributes as important when they are actually not. Therefore, we added a function that checks for falsely detected important attribute (Line 21). In more detail, an important attribute $att$ is confirmed if the following condition is true for each $lr \in LR$ concerning the same action and activity: there exist at least one other reward $lr_i$ where: for all other important attributes $att_i \neq att$ the value of the attribute in $lr_i$ is equal to its value in $lr$ and both rewards have different $FD$.

In the example shown in Figure 3, after receiving the second $fr$, the circle is not confirmed as important attribute (in green); the reason is that: for the other important attribute (the square), there is no other reward that holds the same square value with a different reward direction. The square is not confirmed as important attribute as well. After receiving the third $fr$, the circle is confirmed as important attribute because the condition is satisfied (knowing that there is no other important attribute).

After generalizing all non important attributes and confirming the detected important attributes, the algorithm checks the set of learned rewards $LR$ for any redundant rewards (line 22). For each reward $lr_i$ and $lr_j$ in $LR$ where $lr_i$ is included in $lr_j$ ($lr_i$ attributes satisfy the constraint of or equal to attributes in $lr_j$):
- if both $lr_i$ and $lr_j$ have the same FD then $lr_i$ is removed from $LR$ (generalizing),
- if $lr_i$ does not have the same FD of $lr_j$ then $lr_j$ is removed from $LR$ (specializing).

The step of checking for redundancies insures that at each reception of a new $fr$ there is at most one $lr \in LR$ that could match it. In this case lines (7:17) are applied. If no match exists, $fr$ is added as a new learned reward (line 19).

7. Performance, Convergence and Scaling Analysis through Simulation

In this section, we present two experiments to test both learning algorithms. The first uses simulations and evaluates the performance of the robot using the learned reward functions of both algorithms. The second presents a convergence and scaling analysis. In these experiments, the update value function returns the average of both values $fct(v^{lr}, v^{fr}) = average(v^{lr}, v^{fr})$.

7.1. Performance analysis

We first experimented our learning algorithms through simulation. For this simulation, we used the activity of projecting a video to the user (detailed in Section 5 and Example 5).

- The Scenario:
In the experiment, this activity is achieved through 6 phases: room selection (main bedroom, second bedroom, living room, kitchen), video length selection (episode, movie), video type selection (fairy cartoon, fantastic cartoon, science fiction, drama, sports, show, historic, comedy), volume level selection (low, medium, high), brightness level selection (low, medium, high) and finally projecting the video (project). For each phase, actions are made until user satisfaction is achieved, then the next phase starts.

- The Procedure:
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 reaching max number of traces.

The MDP state represents:
- the possible users’ profiles: age (child, teenage, adult) and gender (male, female),
- activity related information: environmental noise (low, medium, high), daytime (morning, noon, afternoon, evening, night), environmental brightness (low, medium, high) and activity phase (room selection, video length selection, video type selection, volume level selection, brightness level selection, projecting).

The MDP set of actions represents all possible robot’s actions. Each possibility for each phase represents an action. For example, there are 4 different possible actions for the room selection phase, 8 possible actions for the video type selection phase. The MDP set of actions in this scenario includes a total of 21 different actions. 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 realizing the activity (moving to the right room, choosing the length of the video,
choosing the video type, setting the sound, setting the brightness and finally projecting). The MDP policy of the first loop of the evaluation respects only the sequence of phases without having any knowledge about users’ preferences. We used Value Iteration Bellman (1957) to calculate the MDP policies at each loop.

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

Each trace simulation was based on a random situation (random value for each profile and environment attribute). The MDP policy is called to choose the best robot action for each activity phase in the scenario. Then, the predefined rules of preference are used to simulate the user feedback over each robot action. A predefined rule concerning the action $a$ is applicable in a situation if the important attributes values of the rule have the same values in the randomly generated situation. Each simulated trace for this scenario includes at least 5 robot actions each followed by its corresponding feedback (in addition to the projecting action at the end of the trace). During one phase, if the chosen action is followed by a negative feedback, other actions are chosen randomly (aside from the MDP policy) until a positive feedback is achieved before passing to the next phase of the scenario (where the MDP policy is used again).

We evaluated both algorithms to learn the reward function on the same randomly generated situations (user profile and activity related information). We followed the procedure described earlier with $n = 100$ traces and $max = 2000$. After each $n$ traces, we calculated the number of the robot’s actions that were followed by a negative user feedback (negative actions). We also compared the number of complete positive traces (i.e. positive experiences), where the robot succeeded to achieve a complete activity with the user without receiving any negative feedback.

- The Results:
Results shown in Figure 4 present the convergence of both learning algorithms to an optimal adaptive and personalized behavior with no negative actions. The first 100 traces were created before any learning process was engaged by selecting random situations followed by random actions. The figure shows that the first 100 traces had 47% negative actions.

![Fig. 4. Comparison between the direct algorithm and the algorithm with generalization: the number of negative actions and the number of positive experiences (with zero negative feedback) presented per 100 experience.](image-url)

Both algorithms were able to learn, at 100%, the predefined rules of preference. Figure 4 shows that the algorithm with generalization creates 10% of negative actions during the second 100 traces and converges to 0% negative actions after 300 Traces. In the second 100 traces, there was 66 completely positive experiences resulting from the algorithm with generalization. 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 second 100 traces and converged after 1000 traces. The second 100 traces included 42 positive experiences only. In general, it is obvious that both algorithms perform better than a random decision making. We notice that the second 100 traces for both algorithms have less negative actions that the first 100 traces which are generated with random actions.

As a first experiment to test our approach, these results prove the interest of detecting important attributes in the interaction situation, as used in the algorithm with generalization, for a better and faster adaptation. Therefore, it seemed necessary to study, through further experiments, the complexity of convergence for the algorithm with generalization (shown in the following of this section). It was also necessary to prove in other experiments that this algorithm is as effective while applied on real users traces as it is while applied on simulated ones. For this reason, we conducted another experiment to analyze the capability of the algorithm with generalization to detect important attributes using real users traces (Karami et al. (2013a)). The experiment with real users was based on a scenario of selecting a menu in a restaurant with the help of RoboWaiter. In Karami et al. (2013a) we also show how the algorithm with generalization handles ambiguities in interaction traces where similar situations (similar profiles and environment settings) lead to different feedback. We also conducted another experiment with real users and a companion robot (EMOX) which is described in Section 8.

### 7.2. Convergence and Scaling Analysis

For the convergence and scaling analysis of our algorithm with generalization, we created a simulated environment to evaluate the importance of each parameter in the convergence complexity.

**- Simulation of traces:**

Similar to our procedure in the first experiment, we simulated interaction traces using some predefined rules or preference to generate the users feedback. However, in this experiment we didn’t use the scenario of projecting a video, therefore, the predefined rules were generated randomly based on a random number of important attributes for each action. The important attributes were then selected randomly and compared later with the detected important attributes by the algorithm with generalization.

The number of the randomly generated rules of preference depends on the number of important attributes for each actions, therefore, it depends on the possible important situations in the scenario $|vAt|^{|imAt|}$. We fixed the number of predefined rules of preference returning a negative and positive feedback as a percentages $\%negFeedRules$ and $\%posFeedRules$ respectively.

**- Parameters:**

The parameters that we took into consideration in the scalability analysis are:

1. the number of attributes (regarding profile and environment) $|At|$,  
2. the maximum number of attributes values $|vAt|$,  
3. the number of robot actions $|A|$,  
4. the maximum number of important attributes per action $|imAt|$,  
5. the percentage of randomly predefined negative rules of preference $\%negFeedRules$,  
6. the percentage of randomly predefined positive rules of preference $\%posFeedRules$.

**- Procedure:**

We used the same 5 steps procedure presented in the first experiment with $n = 100$ and $max = 2000$. After each $n$ traces we counted the number of the robot’s actions that were followed by a negative user feedback (called negative actions in Figures 5 to 9). In this simulation, for each trace, a situation is presented and the robot has to make one positive action to satisfy the user. If the feedback is negative, the robot has the chance to propose 2 more actions. Therefore, each trace has a maximum of one positive action and maximum 3 negative actions.

**- Convergence results:**

We analyze the importance of each parameter in the convergence of our algorithm with generalization (proved to be faster to converge than the direct algorithm Karami et al. (2013b)). In Figure 5, we fixed all parameters and ranged
the number of attributes $|At|$ from 5 to 9 (with $|vAt| = 4$, $|imAt| = 3$, $|A| = 20$, $\%negFeedRules = 6$, and $\%posFeedRules = 2$). Such parameters values represent a state space (profile and environment possible situations) of $4^5 = 1024$ to $4^9 = 262144$. We notice that augmenting the number of attributes to 8 prevent the algorithm to converge with less than 2000 traces. However, the percentage of negative actions does not exceed 10%. Moreover, augmenting the number of attributes to more than 8 leads to a time of treatment of more than 10 minutes for 2000 traces.

We also notice some local convergences. For example, in the line representing $|At| = 7$, we notice the convergence between 400 and 700 traces before re-augmenting for another peak to around 5% of negative actions in the 900 and 1200 traces loops. This is due to confronting different situations from the previous ones, which generally leads to a detection of new important attributes for one of the actions.

In Figure 6, we fixed all parameters and ranged the number of attributes values $|vAt|$ from 4 to 9 representing from 256 to 6561 states (with $|At| = 4$, $|imAt| = 3$, $|A| = 20$, $\%negFeedRules = 6$, and $\%posFeedRules = 2$). We notice that augmenting the number of attributes values to 9 does not prevent the algorithm from converging with less than 2000 traces. Moreover, the percentage of negative actions does not exceeds 10%. In real scenarios where an important number of attributes values is needed (e.g. continuous domains like time), a possible solution is the discretization of the domain of values (e.g. morning, noon, afternoon, ...).

Figure 7 shows results when ranging the number of important attributes from 4 to 7 (with $|At| = 7$, $|vAt| = 3$, $|imAt| \in \{4, \ldots, 9\}$, $|A| = 20$, $\%negFeedRules = 6$, and $\%posFeedRules = 2$). This means that, in the best case, the system will detect 4 to 7 important attributes for each of the possible actions $|A| = 20$. Convergence might be reached before the complete detection of all important attributes. This is because, some actions are optimal in several situations and the system does not fall into more negative situations to learn...
more important attributes. This is mainly dependent on the \%negFeedRules value. The higher this percentage is, the more learning is needed for convergence (see Figure 9). In Figure 7, the line representing \(|imAt| = 7 = |At|\) shows the results when all the representing attributes are considered important for all actions (which is not the case in most real scenarios).

Figure 8 shows that augmenting the number of possible actions to 45 (with \(|At| = 6\), \(|vAt| = 3\), \(|imAt| = 4\), \%negFeedRules = 6, and \%posFeedRules = 2) does not prevent the convergence with maximum 1200 traces.

Augmenting the percentage of randomly predefined positive rules of preference does not augment the complexity of convergence. The main reason is that our method considers a lack of feedback (feedback=0) as a positive feedback, i.e. the robot does not look for another action to reach a strictly positive feedback. Therefore, we concentrate on the percentage of randomly predefined negative rules of preference.

In Figure 9, we show that augmenting the percentage of negative rules of preference to 35% of important situations (35\%\(|vAt|^{imAt}| = 35\times(3^4)/100 = 28\) negative rules over 81 possible rules) for each action (with \(|At| = 6\), \(|vAt| = 3\), \(|imAt| = 4\), \(|A| = 20\), \%negFeedRules \in \{10, 15, 20, \ldots , 35\}\), \%posFeedRules = 6, and \%posFeedRules = 2) prevents the system from converging with less than 2000 traces. As explained earlier, the increase of number of negative rules of preference leads to more experiences with negative feedback and therefore more complexity in choosing a positive action.

We remind that those predefined rules are needed in our traces simulations as a source of users’ feedback. However, in real scenarios, real user’s feedback will be received during real interactions and there will be no need for such predefined rules. In real scenarios, it is unexpected that more than 35% of possible important attributes values combinations represent situations with a negative feedback.

![Percentage of negative actions per 100 experiences](image)

**Fig. 8.** Convergence analysis: \(|At| = 6\), \(|vAt| = 3\), \(|imAt| = 4\), \(|A| \in \{20, 25, 30, \ldots , 45\}\), \%negFeedRules = 6, and \%posFeedRules = 2.

To conclude, we notice that the convergence of our proposed approach is mainly dependent on the number of attributes representing the problem (\(|At|\)) and the complexity of users’ feedback vs. the possible important situations (\%negFeedRules). For problems represented with more than 8 attributes and 4 values for each attribute, the system might converge to an optimal behavior with no less than 2000 interaction traces. However, during those 2000 and more interactions, less than 10% of actions are negative, which we think is acceptable for certain household entertaining scenarios that have no identified risk for users.

These simulated analysis show that scenarios of important representation size can be treated with our algorithm with generalization to lead a companion robot to an adaptive behavior. However, as they are simulated, these analysis do not show the level of acceptance of such behavior by real users especially when the learning phase takes a long time before optimal convergence. For this reason, we present in the following section, an experiment with real users and an analysis of their experiences with the EMOX adaptive robot.

![Percentage of negative actions per 100 experiences](image)

**Fig. 9.** Convergence analysis: \(|At| = 6\), \(|vAt| = 3\), \(|imAt| = 4\), \(|A| = 20\), \%negFeedRules \in \{10, 15, 20, \ldots , 35\}\), and \%posFeedRules = 2.
8. Experiment with EMOX Robot

We ran an experiment with EMOX (Figure 1a) to test our algorithms on a real scenario with a real robot and to experiment users’ approval of EMOX as a companion robot. We chose as a scenario, an adaptive activity selection by the robot.

8.1. The scenario

In this experiment, when a user is identified by the robot, the latter proposes a personalized activity to the user. The robot chooses one of 11 possible activities: call someone, watch TV, play music, remind agenda, listen to news, listen to weather forecast, order a meal, play a game, sport exercise, practice music or cook. During the experiment, only the proposition of the activity was made, we suppose that after the activity is selected, the robot will assist or guide the user through his/her activity (e.g. propose TV shows, give cooking recipes or simply be a supportive sport coach).

8.2. The procedure

The experiment was ran in two phases. The first was for collecting interaction traces to learn from, using the algorithm with generalization. The second phase was to evaluate the learned reward function and the resulted robot behavior. In both phases, all participants filled their profiles (shown in Table 1) before participating in the experiment. As shown in the table, each profile is defined with 7 attributes and each attribute has a domain of 2 values. Therefore, the total number of possible profiles is $2^7 = 128$. Two profiles of two different users are considered similar if they have the same attributes values and considered different else-wise. In the first phase, there were 17 participants with 17 different profiles. In the second phase, there were a total of 16 users, including 9 users from the first phase, and 7 new users (2 of them have new profiles and 5 have profiles similar to other participants from the first phase).

We considered two environmental settings/parameters: the place (kitchen, living room, office, bedroom or bathroom) and time (morning, noon, afternoon, evening or night).

For each experience in the first phase, the robot identifies the user (profile), the place and the time, then asks the user to sort, in his/her order of preference, the activities that s/he would probably like to do in such situation (i.e. positive activities). The robot also asks the user to select the activities that s/he would never think of doing in such situation (i.e. negative activities). A trace is created from each experience where the user feedback obsel is set to -1 for each negative activity and to $(\frac{\text{number of positive activities}}{\text{order} - 1})$ for each positive activity.

All traces from the first phase were used to learn the MDP reward function before running the value iteration Bellman (1957) to calculate the action value function for the adaptive activity selection. The learned reward function is then used to calculate the MDP policy, which is used in the second phase of the experiment.

For each experience in the second phase, the robot identifies the user (profile), the place and the time, then using the learned MDP policy, it proposes the personalized activity with the highest action value knowing the current state (knowing the user profile and the activity related information). If the user accepts the activity, the experience is ended.
If the user refuses the activity, the robot proposes the second best activity. In case of another rejection the robot proposes the third best activity. In this experiment, we stopped at three propositions by the robot. It was not possible for the robot to propose more than 3 activities at each experience to avoid annoying the user. A trace is created from each experience where the user feedback obsel is set to -1 for each refused activity and to +1 for the accepted one (if exists). Traces from second phase experiences were learned from (at run time) during experiences that follows.

8.3. Interaction with EMOX

Currently, the way to communicate with EMOX is through hand movement. The robot, can detect and track a human body and his open hand palm. This way, users were able to communicate with the robot by moving their hand (up, down, right and left) and the robot was able to communicate with users by projecting images on the floor (Figure 10).

Fig. 10. Participants communicating with EMOX.

To detect users/profiles, we chose as a temporary solution that each user select his own profile. Knowing the list of all users names, the robot presents them one by one and with a right or left hand movement by the user, the list slides to the right or the left. When the right name/profile shows, the user can select it by a hand push gesture (approaching the hand towards the robot then back to its initial position).

The same way of sliding and selecting is used during the experiments to select positive and negative activities (first phase) and to accept and refuse a proposed activity (second phase).

8.4. Results and Discussion

The 17 participants of the first phase generated 89 traces that were used to learn the reward function before starting the second phase. During the second phase, the participants generated 68 traces.

In the second phase, we counted (47%, 29%, 9%, and 12%) experiences with respectively (0, 1, 2, and 3) negative propositions (i.e. the robot proposes an activity that is refused by the user). In two experiences of the 68 ones, the robot did not propose any activity, the reason for this is that no proposition had a positive action value using the learned reward function, and therefore, no activity was proposed to the user (example situation, bathroom at noon). We notice that in 47% of the experiences, the first activity proposed by the robot was the adapted activity to the user. We also notice that in 76% of the experiences, the robot’s first or second proposition was a good one knowing that the robot has 11 possible propositions to make.

Regarding the ability to adapt to new users/profiles, we analyzed the 7 experiences with 2 new profiles in the second phase. We counted (1, 5, 0, and 1) experiences with respectively (0, 1, 2, and 3) negative propositions. We notice that the learned reward function was able to generate a policy that is adaptive to new users (most experiences were user satisfying in the second activity propositions).

After analyzing the 68 traces of the second phase, we noticed that 10 out of 11 possible actions (activities) were proposed and accepted in at least 3 traces. The majority of accepted actions were “listen to music” and “play a game” (around 15 each and less than 10 for the other actions).

After the second phase, we asked the participants to fill a questionnaire using Likert scales, regarding their satisfaction towards their experiences and the robot behavior.
1. Do you think the size of the robot is coherent with the service that it proposes?
   
   - Strongly disagree (1)
   - Disagree (1)
   - Neutral (5)
   - Agree (7)
   - Strongly agree (2)

2. Do you think it was easy to communicate with the robot?
   
   - Strongly disagree (0)
   - Disagree (9)
   - Neutral (4)
   - Agree (3)
   - Strongly agree (0)

3. Do you think the way of communicating with the robot is original?
   
   - Strongly disagree (1)
   - Disagree (1)
   - Neutral (5)
   - Agree (7)
   - Strongly agree (3)

4. Do you think there must be other ways of communicating with the robot (ex. vocal, facial expressions, …)?
   
   - Strongly disagree (1)
   - Disagree (0)
   - Neutral (0)
   - Agree (4)
   - Strongly agree (11)

5. Do you think the level of reactivity of the robot is satisfying?
   
   - Strongly disagree (0)
   - Disagree (6)
   - Neutral (4)
   - Agree (6)
   - Strongly agree (0)

6. Do you think a companion robot can be helpful to you?
   
   - Strongly disagree (2)
   - Disagree (2)
   - Neutral (6)
   - Agree (6)
   - Strongly agree (0)

7. Do you think it would be helpful to have a companion such as EMOX at home to help you in your everyday life?
   
   - Strongly disagree (2)
   - Disagree (4)
   - Neutral (6)
   - Agree (3)
   - Strongly agree (1)

8. Generally speaking, do you think having a companion robot would be helpful to other members of your family (children, elderly, …)?
   
   - Strongly disagree (1)
   - Disagree (2)
   - Neutral (1)
   - Agree (10)
   - Strongly agree (2)

9. Do you think a companion robot can add a social presence to your daily life?
   
   - Strongly disagree (1)
   - Disagree (4)
   - Neutral (2)
   - Agree (6)
   - Strongly agree (3)

10. Do you think the proposed activities were satisfying?
    
   - Strongly disagree (0)
   - Disagree (3)
   - Neutral (4)
   - Agree (9)
   - Strongly agree (0)

11. According to you, are the first activities proposed to you by EMOX were convenient?
    
   - Strongly disagree (0)
   - Disagree (1)
   - Neutral (5)
   - Agree (9)
   - Strongly agree (1)

12. Do you think EMOX is able of adapting to different situations (e.g. afternoon in the kitchen, morning in the living room, …)?
    
   - Strongly disagree (2)
   - Disagree (9)
   - Neutral (2)
   - Agree (3)
   - Strongly agree (0)

13. Do you think EMOX is able to adapt to your preferences?
    
   - Strongly disagree (0)
   - Disagree (0)
   - Neutral (3)
   - Agree (12)
   - Strongly agree (1)

14. Did EMOX, in general, proposed to you pertinent activities?
    
   - Strongly disagree (0)
   - Disagree (0)
   - Neutral (3)
   - Agree (12)
   - Strongly agree (1)

15. Do you think EMOX is able to satisfy your needs after proposing several activities?
    
   - Strongly disagree (0)
   - Disagree (1)
   - Neutral (2)
   - Agree (11)
   - Strongly agree (2)

16. Do you think EMOX needs more information about you and the environment for better adaptation?
    
   - Strongly disagree (0)
   - Disagree (2)
   - Neutral (2)
   - Agree (6)
   - Strongly agree (6)

17. I was motivated by the idea of participating in this experience:
    
   - Strongly disagree (0)
   - Disagree (1)
   - Neutral (4)
   - Agree (8)
   - Strongly agree (3)

18. I found my experience with EMOX pleasant:
    
   - Strongly disagree (0)
   - Disagree (1)
   - Neutral (2)
   - Agree (11)
   - Strongly agree (2)

19. I was annoyed during the experience:
    
   - Strongly disagree (2)
   - Disagree (10)
   - Neutral (2)
   - Agree (2)
   - Strongly agree (0)

Table 2: Users’ responses to the questionnaire
Table 2 shows the results of their responses. Although our main interest during this experiment is to test the adaptive robot behavior and not the ways of interaction with the robot, we concerned few questions to the subject of the interaction with the robot. Most of the users agree with the idea of adding new ways of communicating with the robot. They found it difficult and tiring to communicate through hand movement. In their propositions, users argued that vocal communications might be easier than using hand movements. Also, simpler hand gestures were proposed, like a thumb up and thumb down gestures to declare positive and negative opinions on robot propositions. However, most participants were satisfied with the proposed activities and they found them pertinent and adapted to their preferences. Also, they almost all agree that the robot misses more information about themselves and the environment to better adapt. This means that a richer representation of profiles and environment is needed.

8.5. Model complexity analysis

We will use the activity selection scenario (see Section 8.1) to show the model representing complexity. For this scenario, the number of attributes was \(|At| = 9\) (7 profile attributes and 2 activity related attributes). The value domain size of these attributes ranges between 2 values (for profile attributes) and 5 values (for activity related attributes). Therefore, the maximum number of situation attribute values is \(max(vAt) = 5\) and the total number of MDP states representing all possible situations in the scenario is \(|S| = 2^7 + 5^2 = 153\). The number of possible robot actions is \(|A| = 11\). The number of detected important attributes for action \((a_i)\) ranged between 0 and 4 attributes \(|imAt|_{a_i} \in \{0, 1, \ldots, 4\}\). In this experiment, we gathered 89 traces to learn from. The number of traces was enough to learn a good adaptation policy and even adapt to certain unknown profiles and situations. In the light of Figure 7 that shows the scalability analysis of a system with similar complexity (red square), the results of this experiment demonstrate that a system does not need to reach convergence (which needs 900 traces in Figure 7) to behave in an acceptable manner. Using almost 100 traces (10% of needed traces), users show a high level of satisfaction over the adaptive behavior of the robot (as shown in this experiment).

9. Conclusion and Future Work

In this paper, we presented an architecture and two learning algorithms for an adaptive robot that learns users’ preferences using users’ feedback over robot actions. The first algorithm is called the direct one because it does not necessarily exploit the knowledge to adapt to unknown situations. The second is called the algorithm with generalization and is able to adapt to newly encountered situations.

Simulated experiments showed the ability of the robot to learn, adapt and personalize its behavior to its different users using these two algorithms. We also focused on experiments with real users and results show that our algorithm with generalization is more time-efficient for fine-tuning the robot adaptivity capabilities than the direct algorithm. Generally speaking, this algorithm tries to generalize the adaptation knowledge acquired from past interaction experiences between users and robot to first-time users and unknown situations. In more details, we ran an experiment with the EMOX robot of Awabot. Participants of this experiment were generally satisfied with the robot behavior. We also presented convergence study to estimate the needed number of interaction experiences in order to reach a nearly optimal robot behavior. In the following we will discuss several points regarding our current and future work.

In future work, we would like to work on automatically updating/completing users’ profiles by analyzing the interaction experiences and on the possible consequences of such change on the results of our approach. The personalized robot service depends sometimes on the user profile (filled by the user himself). However, this latter is a subject of change with time (e.g. a user starts to play sports more or less than the usual). Such change can be detected by analyzing the interaction experiences that concern this user. This change can be registered automatically to the user profile (with/without his approval). We would like to prove the assumption that, if the robot has a sufficient learning knowledge, it will be able to detect a change in the user profile and to adapt its behavior automatically.

We also think that it is interesting to treat the problem of multi-user activities. Many questions evolves if the adaptive robot should treat activities that share more than one principal user. First, how to create a group profile using individual user’s profiles. Second, how to learn an adaptive behavior
toward the group using the interaction_traces/preferences learned from each individual in the group.

In the direct algorithm, contradictions between the robot learning knowledge and users feedback are treated to detect missing attributes in the problem representation, which is not the case in the algorithm with generalization. We would like to work on the possibility of automatically detecting the need of more information about the activity related information or the user profile while using the algorithm with generalization.

We have proven in our experiments that the detection of the important attributes helps the learning process to converge faster for personalized decision making in multi-user environments. Current state of the art approaches using user feedback lacks consideration for personalized robot behavior. The COBOT chat system proposed in Isbell Jr et al. (2006) concentrates on adaptive behavior in multi-user environments using reinforcement learning. However, as the authors argue in their paper, the problem of learning from multi-users (not experts) feedback has certain properties mainly that the users have different characters and depending on their characters and the application itself, they tend to react in rewarding with positive actions or only penalizing with negative actions. Also, a lot of ambiguities in the interaction traces can result from human mistakes or missing information in the model. Furthermore, the learning process should deal with the confusion caused by the lack of feedback from users. A lack of feedback in some scenarios might represent a satisfaction and in others it might represents an unsatisfying situation. Also, it is important to take into account contradictions in feedback and, as we mentioned earlier, carry the possibility of detecting the lack of information in the representative state space. We are currently concentrating on studying the effectiveness of our proposed generalized algorithm in different framework structures (sequential and non-sequential decisions, penalizing or rewarding user characters, etc.) and comparing with the effectiveness of other methods as contextual bandit algorithms and decision trees based algorithms on each framework property.

Acknowledgments

We thank the members of Awabot robotic company for their assistance while using their EMOX robot. The Awabot robotic company is the head leader of the FUI-RobotPopuli project that financed this work. We also thank the Gamagora class students and the members of Awabot and LIRIS laboratory who participated in the experiments for their time and comments.

Funding

This work is funded by the French ministry of industries through the Unique Inter-Ministry Fund (FUI) RoboPopuli project.

References


Abir B. Karami is currently a contractual assistant professor at the École des Mines in Douai, France. She is a member of the computer science and automation department (http://ia.ur.mines-douai.fr/). Her research interests focus on machine learning and decision making for adaptive systems (companion robots, smart homes, recommendation systems, etc.). She concentrates on learning mechanisms using users’ implicit and explicit feedback.

Karim Sehaba is Associate Professor of Computer Science at the University of Lyon 2, member of the SICAL (http://liris.cnrs.fr/sical/) team of LIRIS CNRS Lab. His main research interest lies in the field of Adaptive Systems. In this context, he is interested in modelling, transforming and visualisation of interaction traces for user behaviour analysis (inter alia : engagements, emotions, …) and sharing experiences between users.

Benoit Encelle is Associate Professor of Computer Science at the University of Lyon 1, member of the SICAL (http://liris.cnrs.fr/sical/) team of LIRIS CNRS Lab. His research interests are mainly focused on Interactive Adaptive Systems, trying to improve System Accessibility, Usability and Utility to users. He is also interested in Human-Computer Interaction and document (re-)engineering, knowledge extraction and reuse on trace-based systems.