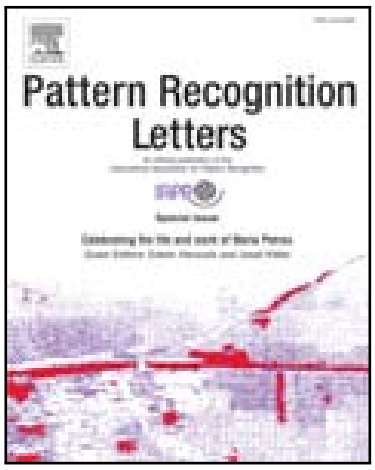


αPOMDP: POMDP-based user-adaptive decision-making for social robots

Gonçalo S. Martins, Hend Al Tair, Luís Santos, Jorge Dias

PII: S0167-8655(18)30082-5
DOI: 10.1016/j.patrec.2018.03.011
Reference: PATREC 7106

To appear in: *Pattern Recognition Letters*

Received date: 23 October 2017
Revised date: 28 February 2018
Accepted date: 12 March 2018

Please cite this article as: Gonçalo S. Martins, Hend Al Tair, Luís Santos, Jorge Dias, αPOMDP: POMDP-based user-adaptive decision-making for social robots, *Pattern Recognition Letters* (2018), doi: 10.1016/j.patrec.2018.03.011

Highlights

- aPOMDP controls an agent's actions to maintain the user in maximum value states.
- Three reward functions based on state value and entropy are proposed and compared.
- Online learning of the transition matrix T is done through a knowledge update step.
- User stays in most valuable states up to 71% of the time, lowering T entropy to 0.7.
- User tests show that the technique is transferable to real scenarios with robots.

Pattern Recognition Letters
journal homepage: www.elsevier.com

αPOMDP: POMDP-based user-adaptive decision-making for social robots

Gonçalo S. Martins[a], Hend Al Tair[b], Luís Santos[a,**], Jorge Dias[a,b]

[a]*Institute of Systems and Robotics, University of Coimbra, 3030-290 Coimbra, Portugal,*
[b]*Khalifa University, Abu Dhabi, UAE*

ABSTRACT

In this work we present αPOMDP: a User-Adaptive Decision-Making technique for social robots. This technique is based on the classical POMDP formulation which we extend with novel aspects inspired by Reward Shaping and Model-Based Reinforcement Learning. Our technique innovates in two main ways: by applying a novel set of rewarding schemes based on the state of the user and by employing a novel execution loop that enables the system to learn the impact of its actions on the user on-the-fly. Our technique has been tested with multiple POMDP solvers and reward formulations in simulations and with real users through the GrowMu social robot. Results show that our technique is able to correctly decide which actions to take, maintaining the user in positive states which interacting with the robot and methodically exploring and learning their characteristics, activities and behaviours.

Keywords: Social Robots, POMDPs, Automated Planning, Decision Making, Machine Learning

1. Introduction

Social and domestic robots aim to assist and accompany the user in their daily life, aiding them in tasks such as keeping track of their medication, suggesting activities and alerting their relatives when emergency situations occur (Martins et al. (2015)). Being a constant presence in the user's life, these systems need to be able to cooperate with them and be accepted. The ability to automatically adapt and adjust to the characteristics of a user can be an important factor in the user's acceptance of the system. This ability is called user-adaptiveness.

As a result, user-adaptive systems, as depicted in Fig. 1, have become a trend in recent research and commercial systems. When fully autonomous, are able to learn or infer the characteristics of their users, building a user model, and make use of that information to inform their following decisions.

In this work we present αPOMDP, a POMDP-based decision-making mechanism able to learn and adapt to a user, as a complement to a pre-existing user model (Martins et al. (2017)). The technique was designed on two main principles: the system should be able to *learn* all of the information it needs to interact properly, and its actions should take into account the *impact* that they produce on the user.

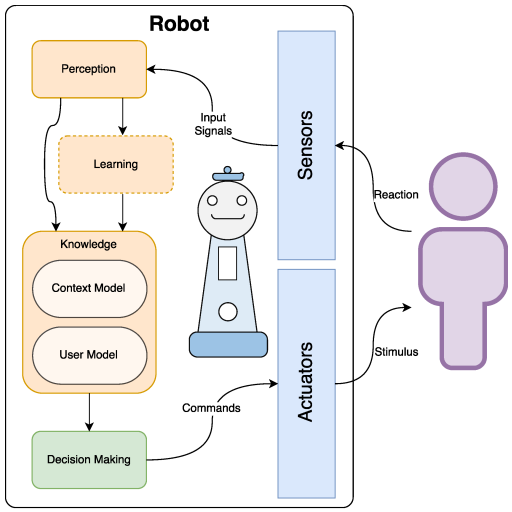


Fig. 1. A general architecture of user-adaptive systems. In this work we focus on the block highlighted in green, decision making.

1.1. Application of POMDPs in Social Robotics

POMDPs are able to model a number of different scenarios, many of which applicable in user-adaptive robots. Their ability to deal with stochastic observations, costs and rewards make them especially suited for decision-making in the uncertain environment of human-robot interaction.

**Corresponding author: Tel.: +351 239 796 389
e-mail: luis@isr.uc.pt (Luís Santos)

This formulation has seen some use in user-adaptive robots, for instance for task allocation in cooperative scenarios (Curran et al. (2016)) or cooperative surveillance (Egorov et al. (2016)).

POMDPs have been used in assistive robots (Taha et al. (2011)), and in the adaptation of a domestic robot to its user's preferences (Karami et al. (2016)). In the latter, the history of interactions is used as basis for the adaptation of future actions.

Beyond the usage of MDPs and POMDPs, user-adaptive robots have been used, for instance, as tutors that can adapt to the pupil's level of skill (Lim et al. (2013)), as learning guides that ultimately become completely autonomous (Sekmen and Challa (2013)), or as therapy companions able to deliver assistance adapted to the user's personality traits (Tapus and Aly (2011); Tapus et al. (2008); Sajid (2016)). Most of these works demonstrate their performance against non-adaptive systems, and establish the need for, and the usefulness of, user-adaptive human-robot-interaction systems.

1.2. Goals and Contributions

The main goal of this work is to present and demonstrate a novel POMDP-based decision-making technique for social robots. This work introduces the following innovative factors:

F1: A novel state-based reward formulation;

F2: A novel learning mechanism and execution loop.

In this work we present experimental results obtained with the proposed system, showing that:

1. The system is able to make decisions that correlate positively with the value functions enconding the impact on the user, keeping them in the most valuable states for significant portions of the experiment;
2. The system is able to learn the impact (Eq.(7)) of its actions on the user, exploring the user's state space to gain information that leads it to improve its performance;
3. The system can achieve these goals in realistic simulations and is transferable to experiments with real users, as shown in the experimental section.

These claims will later be discussed in Section 5.

1.3. Manuscript Structure

Section 2 presents related work, specific goals and claims; Section 3 presents the αPOMDP technique, including the several rewarding schemes we have devised and our transition learning technique. Section 4 presents our experimental methodology, which results are discussed in Section 5. Lastly, Section 6 presents our concluding remarks and future work.

2. Related Work

2.1. POMDPs and Decision Making

The goal of decision making (or automated planning) algorithms is to select an agent's actions such that it achieves a given goal. Generally, the agent receives a set of percepts, such as processed sensory input, and outputs commands to an underlying actuation mechanism (Ghallab et al. (2016)).

POMDPs (Smallwood and Sondik (1973)) are an automated planning technique which differs from its counterparts by taking a probabilistic approach as to the state that the agent is in, i.e. does not assume that the agent knows its current state, expressing its knowledge as probability distributions which are refined as the agent gains information. To compensate for this, the system assumes that the world is static, *i.e.* the world is only assumed to change as a result of the agent's actions. It is assumed that taking an action is an atomic procedure.

Decision-making techniques are domain-independent, in the sense that, given the correct assumptions, they can be applied to artificial agents in different domains with small changes. The key to the successful application of these techniques to different domains lies in the *variable grounding problem*, *i.e.* in the relationship that is established between the problem that the technique is tackling and the real problem at hand. Variable grounding can be achieved by languages such as Planning Domain Description Language (PDDL) (Ghallab et al. (1997)), which bridge the real-world problem and the abstract problem that a decision-making technique solves. Languages such as PDDL aims to model the application domain of a decision-making technique, thus allowing for the grounding of abstract decision making into realistic domains. POMDPs, like other decision-making techniques, are domain-independent.

Alternatively, the problem at hand can be grounded manually, specifying the real-world problem in a way that can be directly tackled by a decision-making technique. We have opted for this approach, since our simplification of the user model allowed for such a direct grounding, as will be seen in Section 3.

2.2. POMDP Definitions

Markov Decision Processes (MDPs) model fully-observable, sequential stochastic decision processes. Within this framework, at each iteration, an agent selects an action to perform based on its policy, which maps its current state to the action it should take. Depending on the current state and action taken, a reward is attributed to the agent, which is used as a basis for optimizing the agent's policy. The overall objective of the agent is to maximize the cumulative reward over the problem horizon.

An MDP is defined as a 4-tuple $< S, A, T, R >$, where S is a finite set of states, A is a finite set of actions, $T(S', S, A) = P(S'|S, A)$ is the probability that a certain action a leads from state s to state s' and $R(S, A)$ is the reward obtained from taking action a in state s. Partially Observable Markov Decision Processes (POMDPs) are an extension to MDPs in which the current s is unknown. They are represented as a 6-tuple $< S, A, T, R, \gamma, \Omega, O >$ (Sondik (1978); Kaelbling et al. (1998)) where S, A , T and R are defined as previously, Ω is a finite set of observations, $O(s', a, o) = P(o|s', a)$ is the observation function, and γ is the discount factor ($\gamma \in [0, 1]$ per time step).

Not being able to observe its state, the agent maintains a belief state $b \in B$, defining the probability of being in state s. b is updated after taking action a and receiving observation o:

$$b^{a,o}(s') = \frac{O(s', a, o) \sum_{s \in s} T(s, a, s') b(s)}{P(o|a, b)}, \tag{1}$$

2.3. POMDP Solving Methods

At each time step, the agent updates its belief $b(s)$. The value function for a single-objective POMDP, V_b, is defined in terms of this belief and can be represented by a set $\mathcal{A}$ of α-vectors.

Each vector α (of length $|S|$) gives a value for each state s. The value of a belief b given $\mathcal{A}$ is:

$$V_b = max_{\alpha \in \mathcal{A}} b \cdot \alpha \tag{2}$$

If an action associated with an alpha vector maximizes the inner product $\alpha \cdot b$, then that action is optimal. Since each α-vector is associated with an action. Therefore, a set of α-vectors $\mathcal{A}$ also provides a policy π that for each belief takes the maximizing action. The optimal value function for a POMDP can be approximated by a piecewise-linear convex function.

Dynamic Programming (DP) is used to solve sequential decision making processes such as POMDP problems (Cassandra et al. (1997)). In POMDP an agent takes the sequencial decisions in a way that maximizes its utility given the actions and the current states. Algorithms to solve POMDP uses various methodologies such as: value iteration (Sawaki and Ichikawa (1978); Cassandra et al. (1994)), policy iteration (Sondik (1978)), accelerated value iteration (White III and Scherer (1989)), structured representation (Boutilier and Poole (1996)), and approximation (Zhang and Liu (1996)). Under each of these methods there are different techniques. For instance, Point-Based methods, such as Point-Based Value Iteration (Pineau et al. (2003)) and Heuristic Search Value Iteration (Smith and Simmons (2012)), have received recent attention for their ability to solve relatively large problems. Q_{MDP}, A simple approximation method, uses the state-action value function $Q(s, a)$ to approximate the alpha vectors (Cassandra and Kaelbling (2016)).

DP consists of number steps yet the most important one is the updating. The update is when new value function V' is defined when value function V is given. The value function is the mapping of information states $R_o^a(s')$ to expected accumulative discounted reward.

$$R_o^a(s') = \frac{Pr(o|s', a) \sum_{s \in S} Pr(s'|s, a) x(s)}{Pr(o|x, a)} \tag{3}$$

$$V'(x) = max_{a \in A}\left(\sum_{s \in S} r_o^a(s) x(s) + \gamma \sum_{o \in O} Pr(o|x, a) V(x_o^a)\right) \tag{4}$$

There are different ways to do the update; one pass (Smallwood and Sondik (1973)), exhaustive (Monahan (1982)), linear support (Cheng (1988)), witness (Littman et al. (1995)), and incremental prunning (Zhang and Liu (1996)).

2.4. POMDP Solvers

Several software packages have been developed that implement POMDP solvers, such as QMDP and SARSOP, which are employed in our experiments. QMDP[1] is an approach to find Q functions for POMDPs by making use of Q values of the underlying MDP $Q(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V(s')$, and linearizing across $Q-$values to obtain the value at a belief: $V(b) = max_{a \in A} \sum_{s \in S} b(s) Q(s, a)$ (Cassandra and Kaelbling (2016)). The Q function for action a, $Q_a(b)$ is the expected reward for a policy that starts in belief state b, takes action a and then behaves optimally. By choosing the action that has the largest Q value for a given belief state, an agent can behave optimally.

Successive Approximations of the Reachable Space under Optimal Policies (SARSOP)[2] (Kurniawati et al. (2008)) is a point-based algorithm. It samples a set of points from the belief space. The sampled points form a tree where the root is the initial belief b_0 which leads to actions and each action leads to observations. The belief tree is refered here as $\mathcal{T}_\mathcal{R}$. Each node of the $\mathcal{T}_\mathcal{R}$ represents a sampled point b. To sample new point b', a node b is selected from the $\mathcal{T}_\mathcal{R}$ as well an action $a \in \mathcal{A}$ and an observation $o \in \mathcal{O}$ according to suitable probability distributions or heuristics. The b' then computed $b'(s') = \tau(b, a, o) = \eta O(s', a, o) \sum_s T(s, a, s') b(s)$ where η is a normalization constant. If all possible sequences of actions and observations are applied than the set of nodes in $\mathcal{T}_\mathcal{R}$ will be exactly the reachable space $\mathcal{R}$. SARSOP avoids this by focusing on finding the approximate cover, maintaining both a lower bound and upper bound on the optimal value function $V*$. It gradually reduces the gap between the upper and the lower bounds on the value function at b_0, until it reaches either a pre-specified gap size or the time limit.

2.5. POMDP Rewarding Schemes

There are different schemes that has been proposed by researchers to enhance the POMDP for various purposes. The *classical* scheme consists of defining a function $r(s, a) : \mathbb{R}^n \rightarrow \mathbb{R}$ that maps each combination of state and possible actions with a scalar reward. This rewarding scheme allows for the modelling of single-objective systems operating under uncertainty.

However, it is possible that a system has to simultaneously optimize multiple objectives, *e.g.* fulfilling its main task while not moving into a certain area of the workspace. In this case, a multi-objective reward function $R(s, a) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ (Soh and Demiris (2011)) can be used. *Multi-objective* rewards encode each objective as a discrete function, which can be optimized separately or through a scalarization scheme:

$$r(s, a) = \sum_k w_k R(s, a)_k \tag{5}$$

thus obtaining a single reward function that is the sum of the multiple rewards, weighted by the w_k weights. *Multi-objective* rewards can also be solved by prioritizing reward functions (Wray and Zilberstein (2015)). In this case, priorities as a set of objectives were proposed using the goal programming lexicographic method for selecting reward preferences.

The POMDP formulation takes into account the belief term b, a probability distribution over the state space encoding the system's belief as to what could be the current state. Previous approaches have attempted to encourage agents to improve their knowledge of the current state via a belief-based reward function (Araya et al. (2010)), reducing the uncertainty in the agent's belief through the use of the entropy Shannon (1948) of the belief state distribution b as a measure of uncertainty.

[1] `https://github.com/JuliaPOMDP/QMDP.jl`

[2] `https://github.com/JuliaPOMDP/SARSOP.jl`

Hybrid schemes can be obtained by combining these approaches, as in Eck and Soh (2012). The reward is weighted by both the system's belief and its entropy, allowing it to gain information on its current state while completing its task.

2.6. Reinforcement Learning and Reward Shaping

Two techniques in the field of automated planning are the closes to αPOMDPin terms of goals and theoretical formulation: reward shaping (Ng et al. (1999)), the basis for factor F1, and model-based reinforcement learning (Szita and Szepesvári (2010)), the basis for innovative factor F2.

Model-based reinforcement learning (MBRL) consists of applying reinforcement learning (Barber (2012)) to a previously-built model. MBRL allows for the fitting of a previous model, such as the transition model $P(s'|s,a)$, to the characteristics of the specific problem at hand. We employ a simplified version of MBRL based on Bayesian Learning in order to adjust our system's transistion model as it executes.

Reward shaping consists of manipulating the reward given to reward-based systems, such as POMDPs, to aid the learning system to achieve the goal. It generally constitutes an informed deviation from the parameters of the problem at hand, heuristically providing a reward that better steers the system towards its goal. We employ an entropy-based reward shaping mechanism that encourages the system to explore potentially rewarding actions in uncertain states, as seen in Section 3.

3. αPOMDP

3.1. Overview

Given that each action of the robot produces an impact on the user, we have designed a system with two basic goals:

- The robot should be rewarded according to the potential impact it produces on the user (F1);
- The robot should be able to autonomously learn the impact of its actions on the user (F2).

In order to achieve these goals, we propose the αPOMDP system (Fig. 2). It extends the regular operation loop of POMDP-based systems with a knowledge integration and policy recalculation steps, which allow the system to gradually learn its impact on the user. Coupled with our novel reward formulations, this system is able to achieve our goals, gaining information as it interacts and steers the user towards valuable states.

3.2. Reward Function

The user's state is modelled as a combination of discrete variables, $s \in \mathbb{N}^n$, where n is the number of user characteristics under analysis, with each variable encoding one of the user's characteristics. A better insight on these characteristics can be found in (Martins et al. (2017)), or inferred through the system's observations as in the original POMDP formulation.

Whenever the system takes an action, it produces an impact on the user under the form of a state transition:

$$s' = \Gamma(s,a), \tag{6}$$

where s' is the user's final state after action a, s is the initial state and Γ is the hidden function through which the user transitions from state to state.

From the user's perspective, s' may be *more valuable*, *less valuable* or *equally valuable* when compared to s. This state value can be represented as a state value function $V(s) : s \to \mathbb{R}$, mapping each possible state to a scalar. This function can encode the *semantic* value of each state, such as the user's health in each state, or their immediate happiness. Thus, it allows us to linearly quantify the importance of each state.

This function allows us to define the impact on the user produced by a robot's action as they transition from s to s':

$$I = V(s') - V(s) = V(\Gamma(s,a)) - V(s) \tag{7}$$

thus quantifying the value of each state transition and, by extension, each possible action.

However, the $\Gamma(s,a)$ function is not known *a priori*, and may even depend on hidden variables. The POMDP formulation already contains a solution to this problem in the form of the $T(s',s,a) = P(s'|s,a)$ function, which encodes state transitions as probability distributions. Thus, in order to make its decisions while taking into account the likely impact on the user, the system can employ T: if an action is unlikely to produce positive impact, then it should be penalized, and vice-versa. As such, the basic reward function (the State Value Reward - SVR) is formulated as follows:

$$\begin{aligned} R(a,s) &= \sum_{s' \in S} T(s,s',a) * I \\ &= \sum_{s' \in S} P(s'|s,a) * (V(s') - V(s)) \end{aligned} \tag{8}$$

resulting in a scalar reward for each possible action in a given state, according to its probable impact on the user. This differs from the classical POMDP formulation, in the fundamental sense that, through the $V(s)$ function, the value of the robot's action is now dependent on the user's states, and not on actions in a given state. Actions are, thus, valued only by their influence on the user, and by their likelihood of transitioning the user to a state that is considered more valuable than the one they are currently in.

In order to encourage the system to gain information on the user, avoiding being stuck in the same state-loop indefinitely, we have devised an information-based term:

$$H = h(T(s,s',a)) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i), \tag{9}$$

where h is the entropy function (Shannon (1948)). The information term H will increase with the uncertainty in the $P(s'|s,a)$ distribution, reading its maximum when the distribution is uniform, *i.e.* when no information on the respective potential transition is known. Thus we formulate the State Value Reward with Information Term (ISVR):

$$R(a,s) = \sum_{s' \in S} T(s,s',a) \cdot I + H \tag{10}$$

By using the information term H, we increase the reward given to an action that leads to unknown transitions, thus encouraging the system to investigate the impact on the user of new actions.

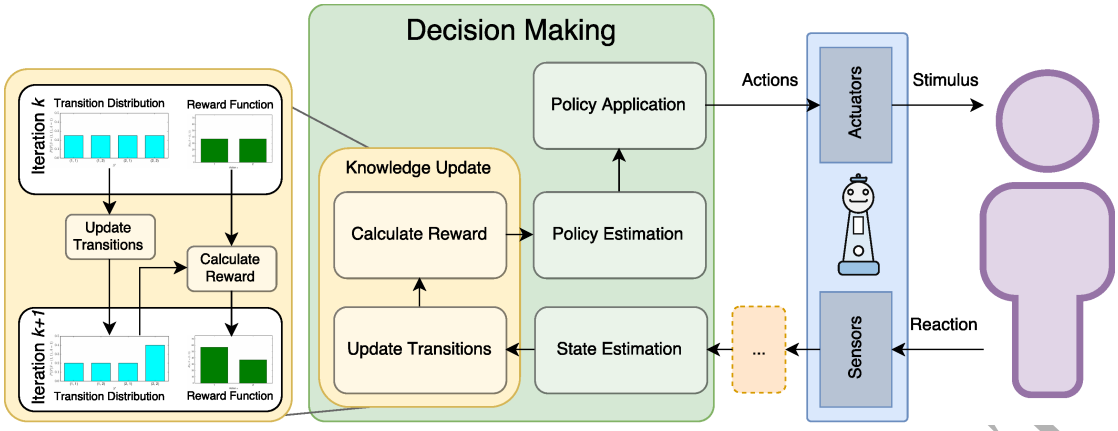


Fig. 2. An expanded view of the "Decision Making" block of Fig. 1, providing an overview of the execution loop of our system. The transition and reward functions can be re-calculated each time new information is obtained by the system, resulting in a new policy which is better suited to the user.

Several $V(s)$ functions can be used to encode multiple rewards, such as when the robot needs to maximize several semantic dimensions of the user, *e.g.* health and happiness. This results in multiple formulations of the SVR of Eq. (8). By scalarizing these multiple objectives we obtain the Multiple State Value Reward (MSVR):

$$R(a,s) = \sum_k \sum_{s'} w_k \cdot P(s'|s,a) \cdot (V_k(s') - V_k(s)) + H \quad (11)$$

which allows for the encoding of different semantic information, such as health status vs immediate happiness, into the weighted reward function through the w_k weights.

3.3. Transition and Reward Learning

In order to predict the impact of its actions on the user, the robot must approximate the Γ function (Eq. (6)). Through the application of the POMDP formulation, Γ can be approximated by $P(s'|s,a)$, which must be learned as the system executes.

Each interaction with the user, as seen in Fig. 2, yields transition information in the form of a sample:

$$L = \{s', s, a\}, \quad (12)$$

a tuple encoding the initial and final states, as well as the action employed by the robot. This information is used to learn $P(s'|s,a)$ by constructing a histogram, as usually seen in the Naïve Bayes Classifier formalism (Ferreira and Dias (2014)):

$$P(s'|s,a) = \frac{1}{N} N(s', s, a) \quad (13)$$

where N is the number of available samples, and $N(s', s, a)$ is the number of samples where $S' = s', S = s, A = a$. A practical example of this mechanism in action is illustrated in Fig. 2.

These tuples are added to the distribution on every interaction, thus enriching the system's knowledge of Γ. By re-calculating the reward functions that depend on this function, the system's information is fully updated. This updated information can then be used to re-calculate the policy, resulting in a policy that is potentially better adapted to the user.

4. Experimental Design

To validate αPOMDP, we applied it in the context of a realist use-case scenario, aiming to demonstrate the claims of Section 1.2 The scenario consists on a user interaction with a social robot, which models the user's state and performs actions to influence it. The robot's actions are assumed to be the only influence on the user's state, and are guided by the αPOMDP technique. We have performed both simulated and tests with real users,with types of experiments being set in the same general scenario. We have released our code as an open-source package containing the code used for these experiments[3].

4.1. Use Case Scenario

Interaction takes place iteratively, following the loop of Fig. 1. Every iteration, the robot should decide about the action to execute, which is dependent on the current state of the user, which is expected to be influenced by the action. For instance, giving the user chocolate could make them happier but potentially harm their health, and performing exercise will lessen their happiness but contribute to better health.

We define the state space as $S : s \in \{S_1, ..., S_5\}, S_i \in \{1, 2, 3\}$ where:

S_1: (User) satisfaction, $S_1 = 1$ means the user is unsatisfied, and $S_1 = 3$ means they are fully satisfied;
S_2: (Robot) robot's current speaking volume, with $S_2 = 3$ meaning that the robot is at full volume;
S_3: (Robot) the robot's current distance to the user, with $S_3 = 3$ meaning that the robot is as close as possible.
S_4: (User) health, with higher levels indicating better health.;
S_5: (World) Time of Day.

With respect to the robot's action space, we define it as $A : a \in \{A_1, ..., A_8\}$ with each action corresponding to:

A_1: Ask the user a question;
A_2: Move the robot 15cm forward;
A_3: Move the robot 15cm back;
A_4: Increase speaking volume by one interval;

[3] https://github.com/gondsm/apomdp

A_5: Decrease speaking volume by one interval;
A_6: Give the user food;
A_7: Give the user candy;
A_8: Do nothing.

4.1.1. Simulation Scenario Variant

Each simulated *trial* consists on the execution of the loop described in Fig. 2 for a set number of *iterations* (n). For each trial, a number of parameters are variable:

1. The POMDP solver in use by the system;
2. The number of iterations that the system will run (n);
3. The reward function to use;
4. The $V(s)$ function(s);
5. The simulated user's profile;
6. The policy calculation period t_c;
7. The state and action spaces.

In order to determine if different solvers have an impact in the system's performance, we have used two different POMDP solvers, QMDP and SARSOP, discussed in Section 2. The remaining parameters are described in detail in the following paragraphs. Each experimental condition was repeated 1000 times for statistical significance.

The system interacted with a simulated user characterized by a $V(s)$ function and a user profile composed of a deterministic Γ function (Eq. (6)) that maps each a and s pair to the resulting s'. Both of these functions were randomized between each *trial*, effectively exposing the system to a new user at each new trial.

The $V(s)$ function was re-generated randomly for each trial, attributing a random discrete value to each state. Similarly, the user profile was re-defined between trials, with each state-action pair being attributed a random destination state. This stochastic generation of the user profile can lead to a number of pitfalls, for instance when the profile is generated in a way that high-value states are unreachable, or that all states receive extremely low values, hindering the system's efforts.

The policy calculation period t_c is the periodicity in which the system is allowed to recalculate its policy, with a $t_c = 1$ meaning that the system re-calculates the policy every iteration. This parameter also varied, deterministically, between tests, namely to determine the system's robustness to the integration of more data, and to allow determining whether the policy needs to be re-calculated at each step for the system to execute successfully.

We have tested all of our three proposals (SVR, ISVR and MSVR). For the MSVR reward, the function was randomized at each trial, with new weights being generated and three simultaneous $V(s)$ functions being used.

Simulation results were obtained using both the full and a reduced version of the scenario of Section 4.1, which limited it to $S = \{S_1, S_2\}$ and three actions. The reduced tests allowed us to select the optimal experimental parameters, which were then applied to the full problem. Interested readers are strongly encouraged to analyze our code and replicate these experiments.

4.1.2. Real Scenario Variant

The real scenario variant is a transposition of the scenario described in Section 4.1 to a real setting, implemented on the GrowMu social robot (Martins et al. (2015)). The state and action spaces were trimmed to variables S_1 through S_3 and A_1 through A_5, limiting our tests to a duration of about 10 minutes. The user's state was estimated at the end of each iteration via verbal interaction. The Human-Robot Interaction occurred as described in Section 4.1, where each iteration consists of the robot performing an action and estimating the resulting user status. Based on the result analysis from the simulated scenario variant trials, we have used the QMDP solver, the ISVR reward, a t_c of 1 and formulated $V(s) = 10 * s_1$, reinforcing only the user's satisfaction.

4.2. Evaluation Metrics

We employ the following evaluation metrics:

R_c: Cumulative Reward;
t_3: Iterations that the system spent in top 3 states;
$\bar{H}(T)$: Average entropy on the $T(s', s, a)$ function;
t: Execution Time.

R_c is defined as

$$R_c = \sum_k R_k \tag{14}$$

where R_k is the reward received by the system up to iteration k. The metric represented by t_3 is defined as

$$t_3 = \sum_{i=1}^{3} N_i \tag{15}$$

where N_i is the number of iterations spent in the i-th most valuable states, as defined by the $V(s)$ function. The average entropy of the T function, $\bar{H}(T)$ is defined as

$$\bar{H}(T) = \frac{1}{N} \sum_{s' \in S} \sum_{s \in S} \sum_{a \in A} H(T(s', s, a)) \tag{16}$$

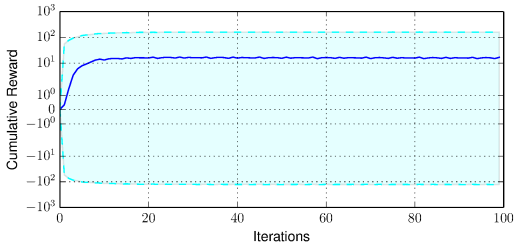
with $H(T(s', s, a))$ defined as in Eq. (9), and N as the number of combinations of s', s and a. Execution time is measured in seconds per trial, and is treated statistically in Section 5.

5. Experimental Results and Discussion

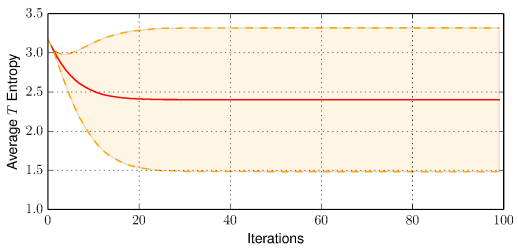
Figures 3, 4, 5 and 6 represent the evolution of the cumulative reward, average T entropy, as well as the count of iterations spent in each state according to its rank, for our simulated trials. The top graph of each figure represents the evolution of cumulative reward for the number of iterations used. The dark blue line represents the average value, while the cyan background represents the $\mu \pm 2\sigma$ area. The middle graph represents the evolution of the average T entropy, $\bar{H}(T)$, metric, with the lighter background representing the $\mu \pm 2\sigma$ area. The bottom graph represents the average number of iterations that the simulated user spent on each state, according to the rank of the state. States are ranked according to their value (as specified by the $V(s)$ function), with state 1 corresponding to the highest-value state, 2 to the second-highest, and so on. This allows us to visualize the system's ability to keep the user in a valuable state since, as mentioned before, each trial uses a different $V(s)$ function and user profile. Fig. 8 presents the results of seven trials that took place in the human scenario, using the same measurements.

Table 1. n **stands for the number of iterations per trial,** t_c **stands for the policy re-calculation period,** $t_{V(s)}$ **stands for the** $V(s)$ **change period,** R_c **stands for cumulative reward,** t **stands for execution time,** t_3 **stands for the number of iterations spent in the three most valuable states,** $H(T)$ **is the entropy of the** $P(s'|s,a)$ **distribution. Each condition (row) was repeated 1000 times.**

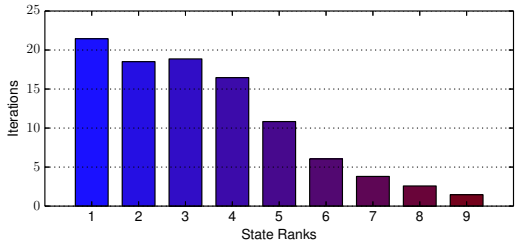
Solver	Reward	n	t_c	Final R_c	t	t_3	Final $H(T)$
QMDP	SVR	100	1	15.164 ± 71.247	1160.456 ± 585.176	58.814% ± 32.752	2.4 ± 0.459
QMDP	SVR	100	20	18.016 ± 73.132	105.912 ± 29.719	51.438% ± 31.714	2.403 ± 0.445
QMDP	ISVR	100	1	96.448 ± 127.955	1477.554 ± 410.262	71.069% ± 19.727	0.697 ± 0.477
QMDP	ISVR	100	20	56.281 ± 111.467	145.815 ± 9.093	35.009% ± 17.185	1.301 ± 0.377
QMDP	MSVR	100	1	92.642 ± 70.935	1579.666 ± 364.688	56.858% ± 24.222	0.439 ± 0.444
QMDP	MSVR	100	20	53.172 ± 55.49	180.573 ± 29.628	35.034% ± 16.29	1.282 ± 0.379
SARSOP	SVR	100	1	17.054 ± 72.459	929.113 ± 181.598	59.318% ± 32.256	2.419 ± 0.459
SARSOP	SVR	100	20	14.797 ± 69.72	86.376 ± 12.59	49.324% ± 30.746	2.379 ± 0.448
SARSOP	ISVR	100	1	100.709 ± 126.883	2246.031 ± 35082.49	70.589% ± 19.313	0.669 ± 0.467
SARSOP	ISVR	100	20	67.367 ± 109.899	88.36 ± 7.483	34.561% ± 16.506	1.283 ± 0.366
SARSOP	MSVR	100	1	91.859 ± 66.424	1183.665 ± 84.448	55.056% ± 25.558	0.425 ± 0.422
SARSOP	MSVR	100	20	55.047 ± 55.485	113.89 ± 47.13	33.687% ± 16.248	1.288 ± 0.372



(a) Evolution of cumulative reward.

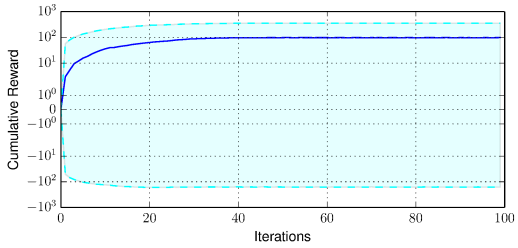


(b) Evolution of T entropy.

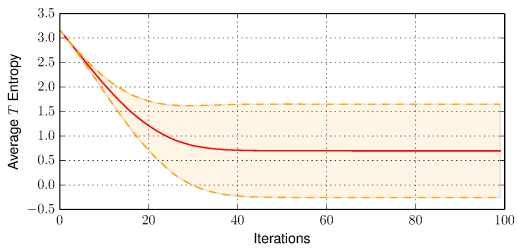


(c) Average iterations spent in each state according to rank.

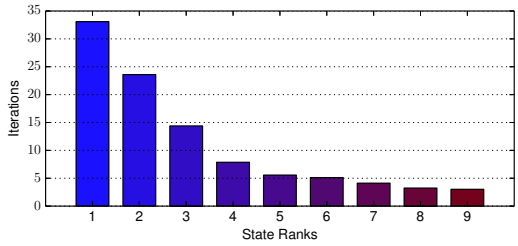
Fig. 3. Results obtained using the QMDP solver and the SVR reward for 1000 trials of 100 iterations, re-calculating the policy every iteration. The top graph represents the cumulative reward, with the colored background representing $\pm 2\sigma$**. Similarly for the middle graph, representing the average entropy in the** $P(s'|s,a)$ **distributions. The bottom graph represents the average number of iterations spent in each state, from the most (left) to least (right) valuable.**



(a) Evolution of cumulative reward.



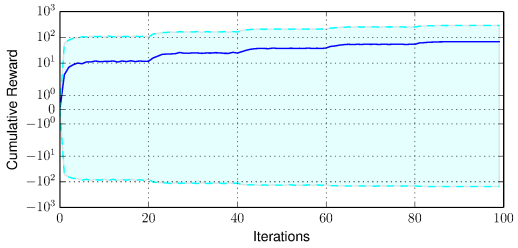
(b) Evolution of T entropy.



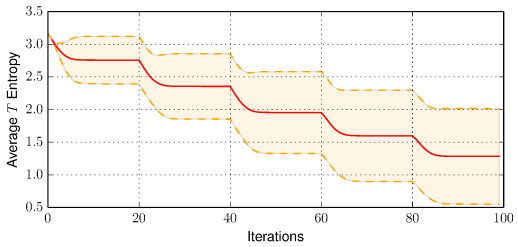
(c) Average iterations spent in each state according to rank.

Fig. 4. Results obtained using the QMDP solver and the ISVR reward for 1000 trials of 100 iterations, re-calculating the policy every iteration. The top graph represents the cumulative reward, with the colored background representing $\pm 2\sigma$**. Similarly for the middle graph, representing the average entropy in the** $P(s'|s,a)$ **distributions. The bottom graph represents the average number of iterations spent in each state, from the most (left) to least (right) valuable.**

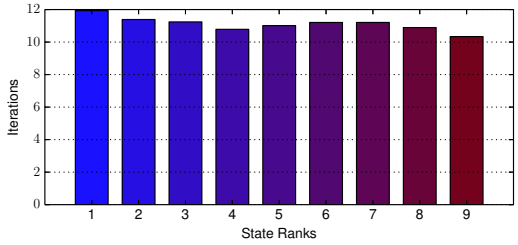
Table 1 presents the aggregate results of our simulated trials for varying conditions. Each row represents a single condition, *i.e.* one combination of the possible input parameters, which was run 1000 times. After the double line, results are presented in the $\mu \pm \sigma$ format, indicating the average and standard deviation for the 1000 trials that took place for each condition.



(a) Evolution of cumulative reward.

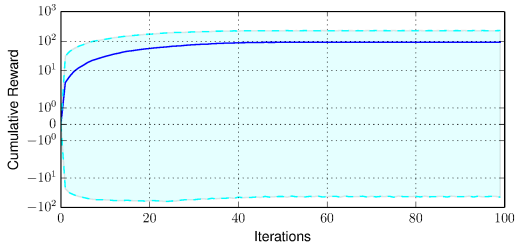


(b) Evolution of T entropy.

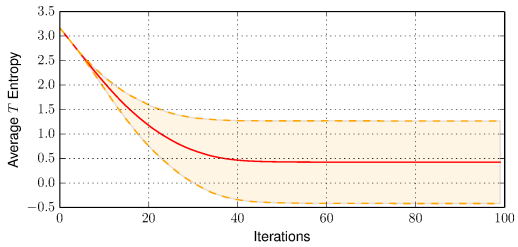


(c) Average iterations spent in each state according to rank.

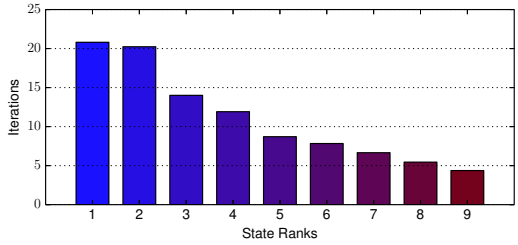
Fig. 5. Results obtained using the SARSOP solver and the ISVR reward for 1000 trials of 100 iterations, re-calculating the policy every 20 iterations. The top graph represents the cumulative reward, with the colored background representing $\pm 2\sigma$. Similarly for the middle graph, representing the average entropy in the $P(s'|s,a)$ distributions. The bottom graph represents the average number of iterations spent in each state, from the most (left) to least (right) valuable.



(a) Evolution of cumulative reward.



(b) Evolution of T entropy.



(c) Average iterations spent in each state according to rank.

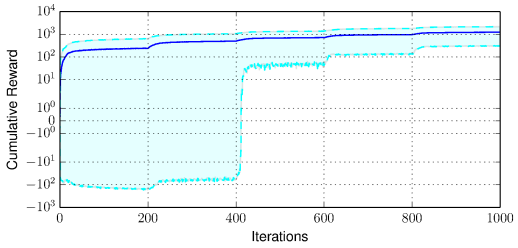
Fig. 6. Results obtained using the SARSOP solver and the MSVR reward for 1000 trials of 100 iterations, re-calculating the policy every iteration. The top graph represents the cumulative reward, with the colored background representing $\pm 2\sigma$. Similarly for the middle graph, representing the average entropy in the $P(s'|s,a)$ distributions. The bottom graph represents the average number of iterations spent in each state, from the most (left) to least (right) valuable.

Table 1 illustrates the results obtained in the simulated trials. We can observe that, in general, the system is able to achieve high cumulative rewards with all rewards. Furthermore, the system is able to maintain the user in the most valuable states, achieving t_3 values of 70% in the best cases. We can also observe that the system is able to obtain low values for the final average entropy of the transition function, reaching values as low as 0.42 bits in the best cases. Thus, in general terms, we can conclude that claim 1 is validated.

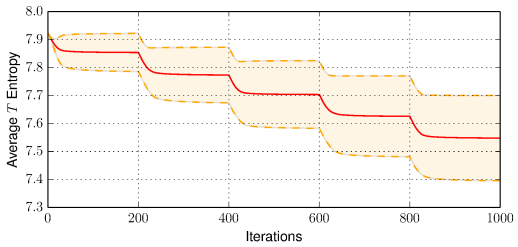
Figs 3 and 4 illustrate the performance of the SVR and ISVR rewards, respectively. We can observe that the ISVR reward obtains, on average, better performance than SVR for all metrics, achieving increases of as much as a 6.3x increase on cumulative reward, 13% increase in t_3 and a 71% decrease in final entropy of the transition function, according to Table 1. This indicates that the ISVR formulation results in a system that is much more capable of maintaining the user in valuable states, and also in gaining information about them. In fact, the ISVR formulation leads the agent to properly explore the user's transitions, thus gaining information that the SVR-enabled agent most likely did not gain. The MSVR formulation, illustrated in Fig 6, performs similarly to ISVR, since it also incorporates the information term that exists in ISVR. Thus, in using the ISVR formulation, our results support claim 2.

Regarding the performance of the solver used, QMDP and SARSOP, we could not find a significant difference. For the R_c, t_3 and entropy metrics, both solvers score similarly to within a 1% difference, meaning that their performance is extremely similar. In terms of average execution time t, however, some larger deviations can be observed, with no solver coming definitely ahead; for instance, QMDP is faster when using SVR, and SARSOP is faster when using ISVR.

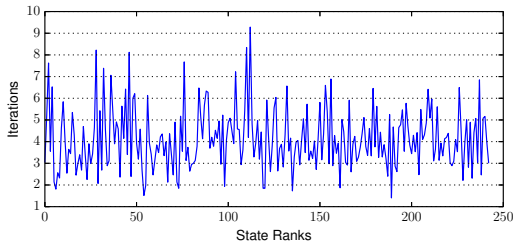
The t_c parameter controls how often the policy is re-calculated. We can observe, in both Table 1 and by comparing Fig 5 to any of the others, that this re-calculation period has a strong impact on all metrics. Firstly, the execution time drops very significantly, to roughly 10% of its original value. However, the remaining performance metrics are mostly im-



(a) Evolution of cumulative reward.

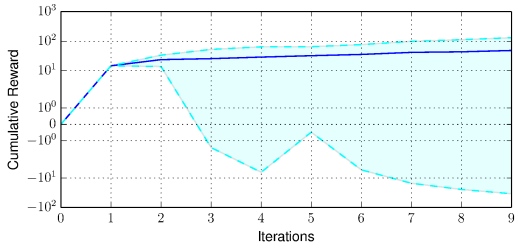


(b) Evolution of T entropy.

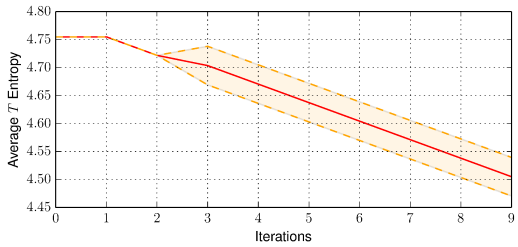


(c) Average iterations spent in each state according to rank.

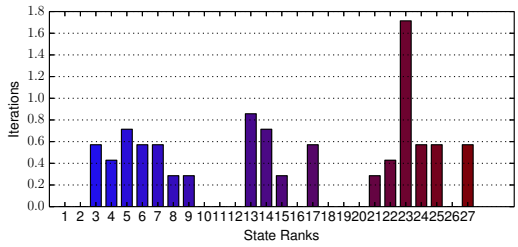
Fig. 7. Results obtained for the complete scenario using the QMDP solver and the ISVR reward for 100 trials of 1000 iterations, re-calculating the policy at every 200 iterations. The top graph represents the cumulative reward, with the colored background representing $\pm 2\sigma$. Similarly for the middle graph, representing the average entropy in the $P(s'|s,a)$ distributions. The bottom graph represents the average number of iterations spent in each state, from the most (left) to least (right) valuable.



(a) Evolution of cumulative reward.



(b) Evolution of T entropy.



(c) Average iterations spent in each state according to rank.

Fig. 8. Results obtained using the QMDP solver and the ISVR reward for trials with human users. The top graph represents the cumulative reward, with the colored background representing $\pm 2\sigma$. Similarly for the middle graph, representing the average entropy in the $P(s'|s,a)$ distributions. The bottom graph represents the average number of iterations spent in each state, from the most (left) to least (right) valuable.

pacted negatively, as seen in Table 1: cumulative reward generally drops to 60% of its original value, as does t_3, with entropy roughly doubling. This can be attributed to the fact that by limiting the re-calculation of the policy, we are effectively limiting the system's ability to integrate new information. However, seeing as the execution time drops very significantly with increases in policy re-calculation, an advantageous tradeoff may be found for different device configurations.

Using this information, we performed the trial of Fig. 7, where the complete scenario of Section 4.1 was tested. We can observe that, while testing with a larger scenario, the techniques performance is largely maintained: its ability to incorporate data is maintained, as observed in Fig. 7(b), as is its ability to converge to a high reward. The system's performance in the t_3 metric suffers with the larger state space, which is explained by the fact that the system was only run for 1000 iterations, which did not allow for a complete convergence to the user profile.

We can observe in Fig. 8 that, similarly to the results obtained in simulation, the system is able to systematically gain information on the user, and gradually increase its cumulative reward. We can also observe that the system was unable to fully maintain the user in the most beneficial states, due to the larger state space used in these experiments, combined with the lower number of iterations allowed. The number of iterations allowed was not enough to achieve full convergence of the learning mechanism and thus the results do not fully correlate with the $V(s)$ function. However, these trials demonstrate that the technique is transferable to real scenarios, thus supporting claim 3, and demonstrate its potential usefulness in long-term scenarios.

6. Conclusion

In this work we have presented and experimentally demonstrated αPOMDP, a User-Adaptive Decision-Making framework based on the POMDP formulation. We have performed tests in a simulated bechmark, demonstrating our technique's abilities while operating on several POMDP solvers, and also with human users, demonstrating its ability to produce impact

on the user. Our results demonstrate the claims from Section 1.2, showing that the system is able to make positive decisions, maintaining the user in valuable states, and also to explore and learn from the user, both in simulated and real trials.

In the future, it would be interesting to extend our testing benchmark, both by applying this system to more complex tasks, but also to combine it with our previous work on User Modelling (Martins et al. (2017)) to achieve a unified user-adaptiveness solution for Social Robots. It would also be interesting to apply our rewarding schemes to non-HRI robotic tasks, such as grasping, wherein our state-value mechanism could be used for reinforcement training of autonomous agents. In the latter case, a domain description language such as PDDL could be used to adapt our technique to the domain in question.

Acknowledgments

This work was developed in the context of the GrowMeUp project, funded by the European Union's Horizon 2020 Research and Innovation Programme - Societal Challenge 1 (DG CONNECT/H) under grant agreement N^{o} 643647.

References

Araya, M., Buffet, O., Thomas, V., Charpillet, F., 2010. A pomdp extension with belief-dependent rewards, in: Advances in Neural Information Processing Systems, pp. 64–72.

Barber, D., 2012. Bayesian Reasoning and Machine Learning. Cambridge University Press.

Boutilier, C., Poole, D., 1996. Computing optimal policies for partially observable decision processes using compact representations, in: Proceedings of the National Conference on Artificial Intelligence, pp. 1168–1175.

Cassandra, A., Littman, M.L., Zhang, N.L., 1997. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes, in: Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc.. pp. 54–61.

Cassandra, A.R., Kaelbling, L.P., 2016. Learning policies for partially observable environments: Scaling up, in: Machine Learning Proceedings 1995: Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9-12 1995, Morgan Kaufmann. p. 362.

Cassandra, A.R., Kaelbling, L.P., Littman, M.L., 1994. Acting optimally in partially observable stochastic domains, in: AAAI, pp. 1023–1028.

Cheng, H.T., 1988. Algorithms for partially observable Markov decision processes. Ph.D. thesis. University of British Columbia.

Curran, W., Bowie, C., Smart, W.D., 2016. Pomdps for risk-aware autonomy, in: 2016 AAAI Fall Symposium Series.

Eck, A., Soh, L.K., 2012. Evaluating pomdp rewards for active perception, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3, International Foundation for Autonomous Agents and Multiagent Systems. pp. 1221–1222.

Egorov, M., Kochenderfer, M.J., Uudmae, J.J., 2016. Target surveillance in adversarial environments using pomdps. Target 15, 20.

Ferreira, J.F., Dias, J., 2014. Probabilistic Approaches for Robotic Perception. Springer International Publishing. URL: https://www.springer.com/us/book/9783319020051, doi:10.1007/978-3-319-02006-8.

Ghallab, M., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D., 1997. PDDL - The Planning Domain Definition Language doi:10.1.1.51.9941.

Ghallab, M., Nau, D., Traverso, P., 2016. Automated Planning and Acting. Cambridge University Press.

Kaelbling, L.P., Littman, M.L., Cassandra, A.R., 1998. Planning and acting in partially observable stochastic domains. Artificial intelligence 101, 99–134.

Karami, A.B., Sehaba, K., Encelle, B., 2016. Adaptive artificial companions learning from users feedback. Adaptive Behavior 24, 69–86. URL: http://adb.sagepub.com/cgi/doi/10.1177/1059712316634062, doi:10.1177/1059712316634062.

Kurniawati, H., Hsu, D., Lee, W.S., 2008. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces., in: Robotics: Science and systems, Zurich, Switzerland.

Lim, G.H., Hong, S.W., Lee, I., Suh, I.H., Beetz, M., 2013. Robot recommender system using affection-based episode ontology for personalization. Proceedings - IEEE International Workshop on Robot and Human Interactive Communication , 155–160doi:10.1109/ROMAN.2013.6628437.

Littman, M.L., Cassandra, A.R., Kaelbling, L.P., 1995. E cient dynamic-programming updates in partially observable markov decision processes. submitted to Operations Research .

Martins, G.S., Santos, L., Dias, J., 2015. The GrowMeUp Project and the Applicability of Action Recognition Techniques, in: Dias, J., Escolano, F., Ezzopardi, G., Marfil, R. (Eds.), Third Workshop on Recognition and Action for Scene Understanding (REACTS), Ruiz de Aloza.

Martins, G.S., Santos, L., Dias, J., 2017. BUM : Bayesian User Model for Distributed Social Robots, in: 26th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN, IEEE.

Monahan, G.E., 1982. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. Management Science 28, 1–16.

Ng, A.Y., Harada, D., Russell, S., 1999. Policy invariance under reward transformations : Theory and application to reward shaping. Sixteenth International Conference on Machine Learning 3, 278–287. doi:10.1.1.48.345.

Pineau, J., Gordon, G., Thrun, S., et al., 2003. Point-based value iteration: An anytime algorithm for pomdps, in: IJCAI, pp. 1025–1032.

Sajid, Q., 2016. Personality-Based Consistent Robot Behavior. Human-Robot Interaction , 635–636doi:10.1109/HRI.2016.7451893.

Sawaki, K., Ichikawa, A., 1978. Optimal control for partially observable markov decision processes over an infinite horizon. Journal of the Operations Research Society of Japan 21, 1–14.

Sekmen, A., Challa, P., 2013. Assessment of adaptive human-robot interactions. Knowledge-Based Systems 42, 49–59. doi:10.1016/j.knosys.2013.01.003.

Shannon, C.E., 1948. A Mathematical Theory of Communication. The Bell System Technical Journal 27, 379–423. URL: http://dx.doi.org/10.1016/S0006-3495(96)79210-X, doi:10.1016/S0006-3495(96)79210-X.

Smallwood, R.D., Sondik, E.J., 1973. The optimal control of partially observable markov processes over a finite horizon. Operations research 21, 1071–1088.

Smith, T., Simmons, R., 2012. Point-based pomdp algorithms: Improved analysis and implementation. arXiv preprint arXiv:1207.1412 .

Soh, H., Demiris, Y., 2011. Evolving policies for multi-reward partially observable markov decision processes (mr-pomdps), in: Proceedings of the 13th annual conference on Genetic and evolutionary computation, ACM. pp. 713–720.

Sondik, E.J., 1978. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. Operations research 26, 282–304.

Szita, I., Szepesvári, C., 2010. Model-based reinforcement learning with nearly tight exploration complexity bounds. ... on Machine Learning (ICML-10) , 1031–1038URL: http://www.icml2010.org/papers/546.pdf.

Taha, T., Miró, J.V., Dissanayake, G., 2011. A pomdp framework for modelling human interaction with assistive robots, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE. pp. 544–549.

Tapus, A., Aly, A., 2011. User adaptable robot behavior. 2011 International Conference on Collaboration Technologies and Systems (CTS) , 165–167URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5928681, doi:10.1109/CTS.2011.5928681.

Tapus, A., Tapus, C., Mataric, M., 2008. User-Robot Personality Matching and Robot Behavior Adaptation for Post-Stroke Rehabilitation Therapy. Intelligent Service Robotics 1, 169–183.

White III, C.C., Scherer, W.T., 1989. Solution procedures for partially observed markov decision processes. Operations Research 37, 791–797.

Wray, K.H., Zilberstein, S., 2015. Multi-objective pomdps with lexicographic reward preferences., in: IJCAI, pp. 1719–1725.

Zhang, N.L., Liu, W., 1996. Planning in stochastic domains: Problem characteristics and approximation. Technical Report. Technical Report HKUST-CS96-31, Hong Kong University of Science and Technology.