The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

αPOMDP: State-Based Decision Making for Personalized Assistive Robots

Hend AlTair,* Gonçalo S. Martins, Luís Santos,** Jorge Dias***
*Robotics Institute, Khalifa University of Science, Technology and Research (KUSTAR), Abu Dhabi, UAE
**AP4ISR Team, Institute of Systems and Robotics, University of Coimbra, 3030-290 Coimbra, Portugal
Email:{hend.altair, jorge.dias}@kustar.ac.ae, {gmartins, luis}@isr.uc.pt

Introduction

Social and domestic robots are aimed at providing assistance and companionship in the daily life of elderly users. In this context, the robot's ability to automatically adapt to the user, user-adaptiveness, can be a deciding factor in the system's success. This work presents αPOMDP, a POMDP-based decision-making mechanism able to learn and adapt to a user, applied to a realistic scenario. Specifically, this work introduces the following key innovative factors:

- A novel state-based reward formulation;
- A novel learning mechanism and execution loop.

Related Work

Partially Observable Markov Decision Processes (POMDPs) are represented as a 6-tuple $< S, A, T, R, \gamma, \Omega, O >$ (Smallwood and Sondik 1973) representing, respectively, the system's states, the agent's actions, the transition and reward functions, the discount factor, the possible observations and the observation model. Not being able to observe its state, the agent maintains a belief state $b \in B$, which defines the probability of being in state s according to the agent's history of actions and observations.

This formulation has been used in user-adaptive robots, for instance for task allocation in cooperative scenarios (Curran, Bowie, and Smart 2016) or cooperative surveillance (Egorov, Kochenderfer, and Uudmae 2016). POMDPs have been used in assistive robots (Taha, Miró, and Dissanayake 2011), and in the adaptation of a domestic robot to its user's preferences (Karami, Sehaba, and Encelle 2016)(Martins et al. 2016).

Several software packages have been developed that implement POMDP solvers, such as QMDP (Cassandra and Kaelbling 2016) or SARSOP (Kurniawati, Hsu, and Lee 2008), of which the former is 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:

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s'), \qquad (1)$$

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

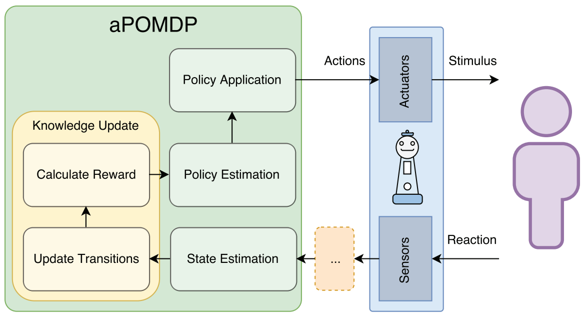


Figure 1: An overview of the execution loop of our system. T and R are be re-calculated when new information is obtained, resulting in a policy which is better suited to the user.

and linearizing across $Q-$values to obtain the value at a belief:

$$V(b) = max_{a \in A} \sum_{s \in S} b(s)Q(s,a). \qquad (2)$$

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.

αPOMDP

We have designed a system based on two basic premises. The robot should:

- Be rewarded according to impact (state change) of its actions on the user ($\rightarrow$ new reward formulation);
- Automatically learn and estimate this impact ($\rightarrow$ new learning mechanism).

The main operational loop of our proposed system is illustrated in Fig. 1. We have extended the regular policy calculation and execution loop of classical POMDPs by introducing a knowledge update and policy re-generation steps. When coupled with our novel reward formulation, this execution loop allows the system to gradually adapt to its user, re-calculating its policy as it gains information as long-term interaction takes place.

Reward Function

Whenever the system produces an action on the user, that action is expected to have an *impact* on the user, which is a direct consequence of the robot's actions. This impact on the user is formulated as a state transition

$$s' = \Gamma(s, a), \tag{3}$$

with s' being the user's final state after action a, s the initial state and Γ a hidden transition function that represents the user's reaction to a certain action in a certain state. The user can attibute subjective value to their current state. As such, whenever the robot produces an action, the user's state may change to a state that they consider to be *more valuable* or *less valuable* than the previous state.

The user's subjective state value is represented as a state value function $V(s) : s \rightarrow \mathbb{R}$. This function allows us to define the impact I on the user produced by a robot's action that causes the user to transition from s to s':

$$I = V(s') - V(s) = V(\Gamma(s, a)) - V(s) \tag{4}$$

In order to modulate each action's reward according to *potential* impact, we make of use of the $T(s', s, a) = P(s'|s, a)$ function. This function, in the classical POMDP formulation, encodes the system's transitions as probability distributions. Coupling it with the I term, we formulate the State Value Reward (SVR) function:

$$\begin{aligned} R(a, s) &= \sum_{s' \in S} T(s, s', a) * I \\ &= \sum_{s' \in S} P(s'|s, a) * (V(s') - V(s)) \end{aligned} \tag{5}$$

Through the $V(s)$ function, actions are valued only by the their potential influence on the user, which when modulated with the T function, allows the agent to make decisions based on the potential impact of its actions. This formulation differs from the traditional POMDP formulation, since it attributes value to states instead of actions contextualized by states.

Transition and Reward Learning

The T function is used in Eq. 5 as an approximation of the user's hidden Γ function. In order to progressively adapt to the user, the system must refine its approximation of this function at each observation of the user's Γ function. Each interaction with the user (Fig. 1) yields one such observation, which is stored in the form of a sample

$$L = \{s', s, a\}, \tag{6}$$

encoding the initial and final states, as well as the action employed by the robot. This information is used to iteratively learn T by constructing a histogram, as usually seen in the Naïve Bayes Classifier formalism, which is updated at every new sample, thus enriching the system's knowledge of Γ:

$$P(S'|S, A) = \frac{1}{N} N(S', S, A) \tag{7}$$

where N is the number of available samples, and $N(S', S, A)$ is the number of samples where $S' = s', S = s, A = a$.

```
Input  : User Profile $user$, number of iterations $n$
Output: Log files.
// Randomize initial state
state = random();
// Determine first policy
policy = solve();
for $n$ iterations do
    // Get action for this state
    a = policy.get_action(state);
    // Get the reward for this action
    reward = user.reward(a, state);
    // Transition state
    new_state = user.transition(state,a);
    // Update the transition function
    update_transition(new_state, state, a);
    // Re-calculate policy and wrap-up
    policy = solve();
    state = new_state;
end
```

Algorithm 1: An illustration of the basic functionality of our HRI simulator.

Experimental Scenario and Set-Up

We aimed to test our technique using a POMDP formulation that approximated a realistic scenario. We have opted for a scenario where the robot has to decide between providing the user with instant gratification and long-term health. On each interaction, the robot must decide which action to take depending on the state of the user. Naturally, each of its actions will have an impact on the user: giving them sweets will provide instant gratification, and performing exercise will contribute to better health. The user and robot interact periodically, with the robot progressively learning the impact of its actions on the user through the learning technique specified before. Thus, the relevant POMDP variables are materialized as:

S: $s \in \{S_1, S_2\}$ where $S_i \in \{1, 2, 3, 4\}$:

- S_1: user satisfaction, $S_1 = 1$ means the user is unsatisfied, and $S_1 = 4$ means they are fully satisfied;
- S_2: user's health, similarly to S_1.

A: $a \in \{1, ..., 3\}$ with each action corresponding to:

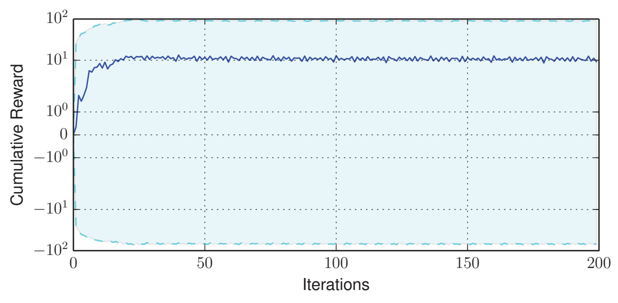
1. Give the user sweets (instant gratification);
2. Encourage physical exercise (long-term benefits);
3. Do nothing.

The remaining parameters are defined as before. For the purposes of this experimentation, the system was given observations such that they matched the real state.

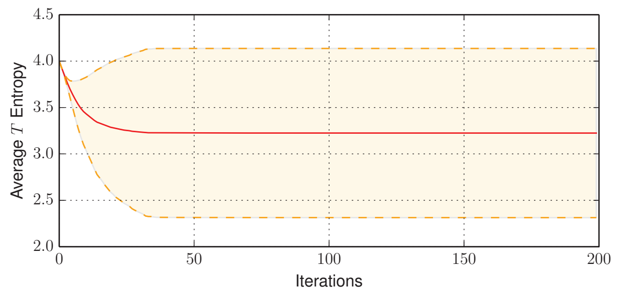
We have developed a simulator that mimics the system of Fig. 1, making use of policies to expose the simulated user to the actions deemed correct by the policy, depending on the current state, as illustrated in Algorithm 1. Each simulated *trial* consisted of executing the loop described in Fig. 1 for a set number of *iterations* (n). The system interacted with a simulated user characterized by:

$$V(s) = 2s_1 + s_2, \tag{8}$$

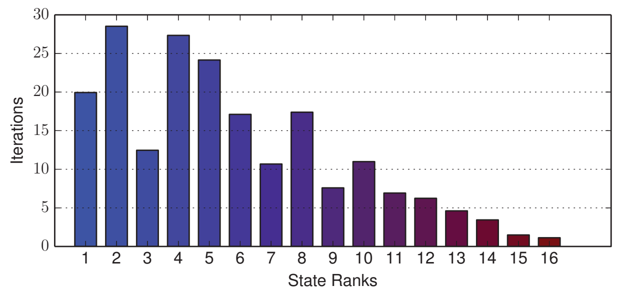
by using these coefficients, we are encoding into the $V(S)$ function the idea that s_1 is more important than s_2, for the



(a) Evolution of cumulative reward.



(b) Evolution of T entropy.

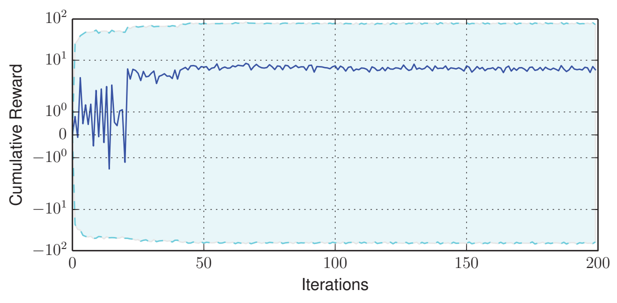


(c) Avg. iterations spent in each state rank.

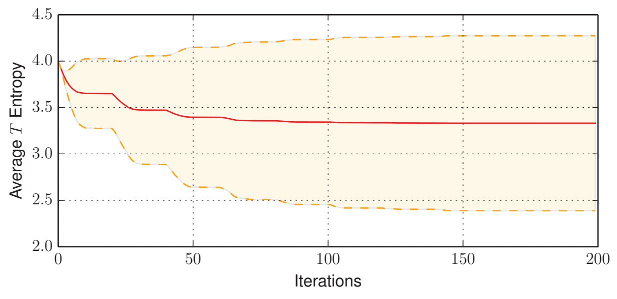
Figure 2: Results obtained for 100 trials of 200 iterations, re-calculating the policy every iteration. (a) represents the cumulative reward, with the colored background representing $\pm 2\sigma$. Similarly for (b), representing the average entropy in the T distributions. (c) represents the average number of iterations spent in each state, from the most (left) to least (right) valuable.

purpose of our tests. A fixed Γ function is used for each trial, and is randomized between trials. As a technological basis for our test-bench, we have used JuliaPOMDP (Egorov et al. 2017), running on Julia 0.6, and the QMDP (Cassandra and Kaelbling 2016) POMDP solver. We have performed trials both solving the POMDP for all possible states every time new information is gathered, and allowing some time between policy re-calculation. We encourage the interested reader to analyze our code and replicate these experiments[2].

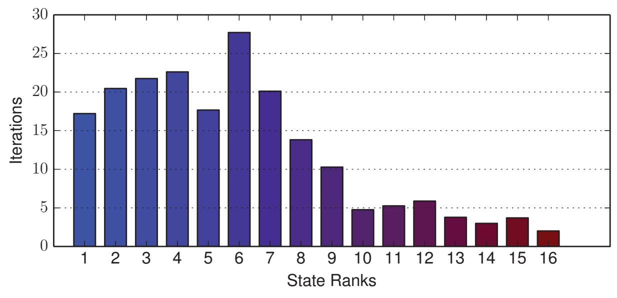
We have evaluated the technique's performance through



(a) Evolution of cumulative reward.



(b) Evolution of T entropy.



(c) Avg. iterations spent in each state rank.

Figure 3: Results obtained for 100 trials of 200 iterations, re-calculating the policy every 20 iterations. (a) represents the cumulative reward, with the colored background representing $\pm 2\sigma$. Similarly for (b), representing the average entropy in the T distributions. (c) represents the average number of iterations spent in each state, from the most (left) to least (right) valuable.

the following metrics:

- R_c: Cumulative Reward;
- $\bar{H}(T)$: Average entropy on the $T(s', s, a)$ function.
- $\bar{t}_n$ Average number of iterations the user spent in each state of rank n, as defined by the $V(S)$ function.

$\bar{H}(T)$ is defined as:

$$\bar{H}(T) = \frac{1}{N} \sum_{s' \in S} \sum_{s \in S} \sum_{a \in A} H(T(s', s, a)) \tag{9}$$

where N is the number of combinations of s', s and a.

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

The entropy function is defined as in Shannon's original formulation (Shannon 1948):

$$H(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i), \tag{10}$$

Results and Discussion

Figures 2, 3 represent the evolution of the cumulative reward, average entropy $\bar{H}(T)$, as well as the count of iterations spent in each state according to its rank. Fig. 2 corresponds to the scenario where the policy was re-calculated for every iteration while Fig. 3 to re-calculation every 20 iterations. The top graph of each figure represents the evolution of cumulative reward for the number of iterations used, with the dark blue line representing the average, while the cyan background represents the $\mu \pm 2\sigma$ area. Similarly, the middle graph represents the evolution of the average T entropy, $\bar{H}(T)$. The bottom graph represents the average number of iterations that the user spent on each state according to the rank of the state, for all trials.

In Fig. 2, the evolution of the reward and T entropy demonstrate that the system is able to progressively learn the user's profile (Γ) and successfully adapt to it, progressively converging and maintaining the user stable in the most valuable states, as seen in Fig. 2c. We can observe that the system goes through a learning phase, from iterations 0 to iteration 30, where most of the user's transition function Γ is captured in the system's user model. In this period, entropy lowers quickly as the system learns, and cumulative reward suffers the highest variations. After this period, the system is able to keep the user transitioning among the most valuable states, as illustrated in Fig 2c.

In Fig. 3 we can observe that the system still achieves the main goals. However, performance is hindered by its reduced ability to incorporate new data: it takes around 100 iterations to reach the same entropy minimum as in the previous test, and the distribution of the user's states in Fig. 3c is leaning significantly more towards lower-value states. The reduced performance observed in this test demonstrates that the system must be allowed to incorporate data frequently, at least in the beginning of the interaction, to ensure that appropriate learning takes place. However, given the advantages of a workflow in which the policy is re-calculated fewer times, namely the potential reductions in computational effort, it is important to consider these results in the design of an adaptive re-calculation loop.

Unlike the classical POMDP formulation, αPOMDP is able to learn the transition matrix T on-line as it interacts with the user, dynamically adapting its rewarding scheme to maximize positive user impact (Eq. 4), factoring in the semantic value of user states in the decision-making process.

Conclusion

In this work we have presented αPOMDP, a User-Adaptive Decision-Making framework based on the POMDP formulation. We have performed simulation tests, demonstrating its ability to learn the impact of its actions, and to maintain the user in valuable states. Results show that the system is able to correctly estimate the impact of its actions on the user, generating policies according to the transitions that it learns while executing, gaining satisfactory cumulative rewards.

Future Work: oPOMDP

As mentioned, we have opted to provide our system with true observations, in order to better evaluate the performance of SVR in optimal conditions. These experiments have allowed us to conclude that SVR is a worthwhile formulation.

In the future, it would be interesting to expand our formulation to remove this limitation, integrating the probability of observation while maintaining the basic tenets of SVR, in a formulation we call oPOMDP. In this formulation we try to estimate the impact of actions in the world as in Eq. 11. Benefits comes with observation estimation is that although implicitly actions might lead to expected observations; explicitly this might not be the case.

$$R^o_{a\in A} = Pr(o|s', a) * \sum_{a\in A} Pr(s'|s, a) \tag{11}$$

References

Cassandra, A. R., and 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*, 362. Morgan Kaufmann.

Curran, W.; Bowie, C.; and Smart, W. D. 2016. Pomdps for risk-aware autonomy. In *2016 AAAI Fall Symposium Series*.

Egorov, M.; Sunberg, Z. N.; Balaban, E.; Wheeler, T. A.; Gupta, J. K.; and Kochenderfer, M. J. 2017. Pomdps. jl: A framework for sequential decision making under uncertainty. *Journal of Machine Learning Research* 18(26):1–5.

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

Karami, A. B.; Sehaba, K.; and Encelle, B. 2016. Adaptive artificial companions learning from users feedback. *Adaptive Behavior* 24(2):69–86.

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

Martins, G. S.; Ferreira, P.; Santos, L.; and Dias, J. 2016. A Context-Aware Adaptability Model for Service Robots. In *IJCAI-2016 Workshop on Autonomous Mobile Service Robots*.

Shannon, C. E. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal* 27(1):379–423.

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable markov processes over a finite horizon. *Operations research* 21(5):1071–1088.

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